中图分类号：TP3
论 文 编 号：10006GS1321F10

北京航空航天大學

# 专 业 硕 士 学 位 论 文

# 面向天文海量数据的数据管理系统设计与实现

作 者 姓 名　何勃亮
学 科 专 业　软件工程
指 导 教 师　王华锋　副教授
培 养 院 系　软件学院

# The Design and Implementation of Data Management System for Massive Astronomical Data

A Dissertation Submitted for the Degree of Master

**Candidate: He Boliang**

**Supervisor: A. Prof. Wang Huafeng**

School of Software

Beihang University, Beijing, China

**中图分类号：TP3**

**论 文 编 号：10006GS1321F10**

**专 业 硕 士 学 位 论 文**

# 面向天文海量数据的数据管理系统设计与实现

作 者 姓 名　何勃亮　　申请学位级别　硕士

指导教师姓名　王华锋　　职　　称　副教授

学 科 专 业　软件工程　　研 究 方 向　大数据技术与应用

学习时间自　2013 年 09 月 10 日　起　　至　2017 年 06 月 04 日止

论文提交日期　2017 年 05 月 15 日　论文答辩日期　2017 年 06 月 04 日

学位授予单位　北京航空航天大学　　学位授予日期　　年　　月　　日

# 关于学位论文的独创性声明

本人郑重声明：所呈交的论文是本人在指导教师指导下独立进行研究工作所取得的成果，论文中有关资料和数据是实事求是的。尽我所知，除文中已经加以标注和致谢外，本论文不包含其他人已经发表或撰写的研究成果，也不包含本人或他人为获得北京航空航天大学或其它教育机构的学位或学历证书而使用过的材料。与我一同工作的同志对研究所做的任何贡献均已在论文中作出了明确的说明。

若有不实之处，本人愿意承担相关法律责任。

学位论文作者签名：_____ 日期：　　年　　月　　日

# 学位论文使用授权书

本人完全同意北京航空航天大学有权使用本学位论文（包括但不限于其印刷版和电子版），使用方式包括但不限于：保留学位论文，按规定向国家有关部门（机构）送交学位论文，以学术交流为目的赠送和交换学位论文，允许学位论文被查阅、借阅和复印，将学位论文的全部或部分内容编入有关数据库进行检索，采用影印、缩印或其他复制手段保存学位论文。

保密学位论文在解密后的使用授权同上。

学位论文作者签名：_____ 日期：　　年　　月　　日

指导教师签名：_____ 日期：　　年　　月　　日

# 摘　要

随着国内近些年来天文学的发展，尤其是天文望远镜的建设，比如郭守敬望远镜(大天区面积光纤光谱望远镜，LAMOST)、五百米口径球面射电望远镜（FAST）等，望远镜的观测数据也逐渐呈现井喷态势，自 2008 年以来，基本以每两年翻一番的规模增长。与此同时，国内外天文望远镜数据的快速增长以及科研范式的发展，也使得天文学领域正式进入海量数据时代，天文数据的管理工作面临着极大的挑战。

本文对天文大数据的管理方面尤其是底层的存储方面的一些关键技术进行了研究，设计并研发了一套分布式存储系统，这是一个专门面向天文海量数据的数据管理系统。在系统设计与实现中：

首先，充分吸收了现有的分布式存储系统的研究成果，比如分布式的架构、RESTful 风格的对象存储模型、海量小文件合并存储策略、索引算法等。

其次，本文针对天文学数据的一些特点进行了深度定制，制定了数据存储模型，比如层次化的存储模型、分布式主程序系统、可小型化或者单机化的部署模式、以及面向不同层次应用的接口等。而且还定义了一批数据归档和管理的规范。

最后，经过模块化的设计，分布式的部署，使整个系统可以满足天文学家、望远镜管理员、科研管理者、公众、乃至程序客户端所需的高性能服务。

系统采用了 Go 语言作为主要开发语言，集成了 KV 数据块、关系型数据库、消息队列等诸多技术，并最终打造了这样一套面向海量天文数据的数据管理系统。基于这样的一套系统，国内天文望远镜数据的管理效率得到了有效的提升。

**关键词：**天文海量数据，虚拟天文台，数据管理，分布式，文件系统

# Abstract

With the development of astronomy in recent years, especially the construction of astronomical telescope, compared with Guo Shoujing Telescope (The Large Sky Area Multi-Object Fiber Spectroscopic Telescope, LAMOST), Five-hundred-meter Aperture Spherical radio Telescope (FAST), etc. Since 2008, the data of observations by Chinese telescope has grown substantially every two years. At the same time, the rapid growth of telescope data at entire world and the development of scientific paradigm also makes the field of astronomy officially entered the Big Data Period. Astronomical data management is facing great challenges.

In this paper, the management of astronomical data is the bottom of the storage aspects of some of the key technologies were studied, designed and implemented a distributed storage system, which is Massive Astronomical Data Management System.

In the system design and implementation:

First of all, fully absorb the existing distributed storage system research results, such as distributed architecture, RESTful style object storage model, multiple copies algorithm, massive small file merge storage, index algorithm, etc.

Secondly, this paper makes a deep customization of some of the features of astronomical data, and develops data storage models such as hierarchical storage models, distributed main program systems, miniaturization or simplification of deployment patterns, and for different levels of application the interface, etc., but also defines a number of data archiving and management specifications.

Finally, after a modular design, the distributed deployment allows the entire system to meet the needs of enough astronomers, telescope administrators, research managers, the public, and even the high-performance services required by the client.

The system uses the Go language as the main development language, integrated KV database, relational database, message queue and many other technologies, and ultimately build such a set of massive astronomical data special for the data management system. Based on such a system, the management efficiency of the domestic telescope data has been effectively improved.

**Key words:** Massive Astronomical Data, Virtual Observatory, Data Management, Distributed, File System

# 目　　录

# 插 图 目 录

# 表 格 目 录

# 术语表

| | |
|---|---|
| FITS | Flexible Image Transport System （FITS），是国际天文学联合会（IAU）1982 年开始制定的在天文界用于数据传输、交换的统一标准格式。 |
| **元数据** | 在天文学数据文件领域，一般泛指 FITS 文件的头信息，它包含了对数据文件的详细描述。 |
| **虚拟天文台** | 虚拟天文台技术是近十多年在国际天文界迅速发展的一系列技术。虚拟天文台（The Virtual Observatory）是利用先进的信息技术将各种天文研究资源以统一的服务模式无缝透明地汇集到一起，形成一个统一的数据密集型的网络化天文研究与科普教育平台。 |
| **中国虚拟天文台** | 中国虚拟天文台（Chinese Virtual Observatory）是中国的虚拟天文台组织，目前已经融合了国内数十台望远镜的数据。 |
| **原始数据** | 原始数据指的是从望远镜观测到的第一手数据，也称作 raw data，一般都包含了大量的仪器信息，需要经过一定的数据处理程序才能转化为供科学家使用的数据。 |
| Pipeline | 天文学领域特指数据处理程序，如将原始数据处理为产品数据的程序。 |
| **产品数据** | 供科学家可以直接使用的数据，一般是经过了 Pipeline 处理过的原始数据。 |

# 第一章　绪论

## 1.1　研究背景及意义

随着近些年中国天文望远镜的发展，国内自产天文数据的规模越来越大，并且呈现井喷状态，以中国科学院国家天文台天文数据中心（中国天文数据中心，CAsDC[1]）为例。2008 年以前，数据中心大多是国外的镜像科学数据，比如美国斯隆数字化巡天数据（SDSS），法国斯特拉斯堡数据中心（CDS）星表数据等。而在 2008 年以后，随着大天区面积多目标光线光谱天文望远镜（郭守敬望远镜，LAMOST）、南极中国小望远镜阵（CSTAR）、丽江 2.4 米望远镜等望远镜开始观测，国产天文数据开始大规模产出，并且以几乎每两年翻一番的速度高速发展，截至 2016 年年底，仅文件型数据的规模就已经达到 110TB。图1是 2008 年以来中国天文数据中心历年国产天文数据规模的情况。

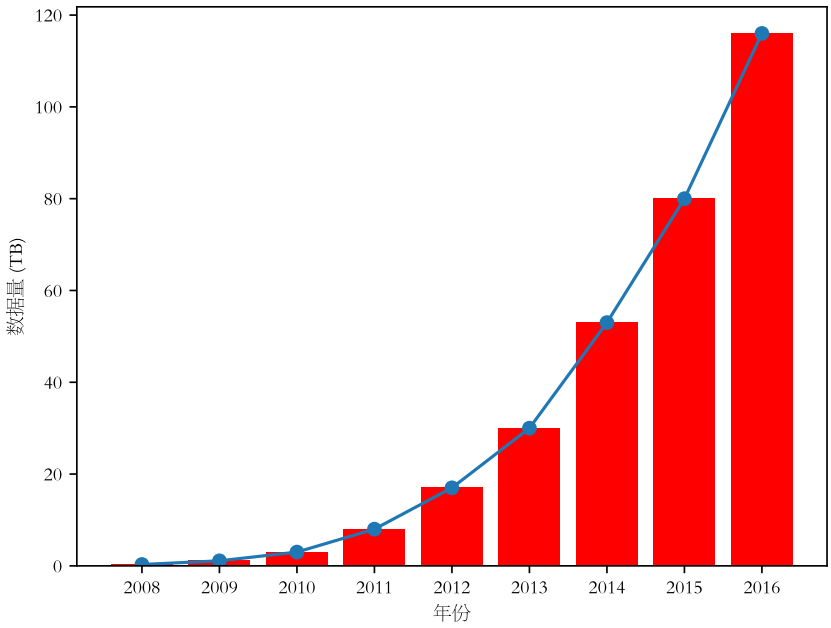


图 1　中国天文数据中心历年数据规模

而在 2016 年 9 月份，随着 500 米口径球面射电望远镜（FAST）的建成，并开始投入试观测，数据的规模将更加巨大。

除了文件型数据，还有大量的结构化数据需要存储到数据库中供科研人员检索，截至到 2016 年年底，部分数据库的单表已经达到 8 亿行左右的规模量级，这对于数据检

[1]http://casdc.china-vo.org

索也带来了不小的压力。随着数据规模而来的，还有数据存储、数据备份、数据访问等的一系列围绕这数据展开的问题。

数据存储方面，目前采用的还是传统的基于文件系统的管理方式，但随着文件数的规模已经达到了上亿量级，文件的访问也成为一个瓶颈。另外，文件型数据的存储未完全考虑数据元信息的收集，比如文件类型、文件大小、文件产生时间等等。

数据备份方面，目前是自动进行的异地备份，只实现了初步的智能化，但和存储一样，也未考虑到数据元信息的收集等。

数据的访问，目前尚无好的方法，使用的依旧是传统的拷贝方法。

基于这些数据管理方面的现状和问题，如何高效地管理数据，并且如何满足未来面向天文海量数据研究领域的数据管理，是个需要进行研究的重要课题。

国家自然科学基金—天文联合基金项目：“天文多节点海量数据归档的关键技术研究（U1531115）”旨在研究天文海量数据存储方面的核心技术，这也是本文的课题来源之一。

总结起来，日益增长的国产天文数据与落后定数据管理方式的矛盾日益突出；在天文海量数据时代，各个层面的用户对数据的需求日益丰富。迫切需要构建一套满足中国天文数据管理与应用的分布式数据管理系统。

## 1.2　国内外天文数据现状

### 1.2.1　国内天文望远镜数据

目前在国内，中国科学院国家天文台数据中心的数据规模是最大的，面临的挑战也是巨大的。中国天文数据中心已经归档的国产数据有：

- 郭守敬望远镜（LAMOST）原始数据、产品数据，数据规模 30TB；
- 南极中国小望远镜阵（CSTAR）原始数据、产品数据，数据规模 2.6TB；
- 南银冠 U 波段巡天（SCUSS）原始数据、产品数据，数据规模 20TB；
- 北京-亚利桑那巡天（BASS）原始数据、产品数据，数据规模 25TB；
- 丽江 2.4 米望远镜数据，数据规模 8TB；
- 南极巡天望远镜（AST3）原始数据、产品数据，数据规模 2TB；
- 丽江 BOOTES-4 望远镜数据，数据规模 5TB；
- 怀柔太阳射电望远镜（SBRS）数据，数据规模 4TB；
- 怀柔太阳磁场望远镜数据，数据规模 200GB；
- 青海德令哈 50cm 双筒望远镜（50-Bin）数据，数据规模 3TB；
- 抚仙湖太阳塔望远镜（NVST）数据，数据规模 1TB；

- 兴隆 2.16 米望远镜数据，数据规模 2TB；
- 其他数据。

这些望远镜分布于全国各地，甚至是南极。数据归档的要求也不大一致，有些只是需要归档，有些需要提供数据服务，还有一些需要为数据的预处理、处理提供服务；有些对数据归档的时效性有要求，有些没有要求等等。

截止 2016 年底，总的数据规模已经达到 110 多 TB。但当前数据归档的方式与归档程序还比较原始，主要采用定时备份的机制进行。数据在产生后的次日定时从观测站点传输至国家天文台数据中心，数据中心再进行二次备份，以保证数据的安全性。

原始数据的元数据库建设也比较初步。比如目前已经建设了的 2.4 米望远镜的元数据库，程序会自动提取原始 FITS 文件的头信息并入库，并记录文件的大小、路径以及校验码等。

国产数据归档还面临着其他一些问题，各个望远镜观测站点缺乏完整的数据管理解决方案，主要还使用比较原始的文件夹方式存储，比如一天一个文件夹存储当天观测的数据等。数据没有审计策略，数据质量的控制比较难以保障。另外各个观测站点的操作系统由于历史原因和使用习惯的原因，也不尽相同，这为构建一套通用的终端数据管理系统带来了一定的难度。

总地来说，当前国产天文数据归档和管理面临的问题主要有：

1. 缺乏站点数据以及基础数据管理。比如无法对站点的数据有一个全面概览的浏览，有多少数据量？都有哪些类型的数据？什么人观测的？等等，这些基础数据都严重缺乏。实际也是数据缺乏全面管理的问题。
2. 缺乏数据节点间数据传输管理。观测站点的数据一般都是通过人工拷贝进行数据的传输，稍微好点的通过网络传输，在传输过程中又有可能出现数据缺失，没有完整性检验，以及传输过程的管理。
3. 无跨平台程序软件系统。各个观测站点的数据存储计算机操作系统不一致，当进行整合管理时，需要针对不同的系统做出不同的管理方案，程序的复杂度加大，管理和维护成本加大。
4. 缺乏对数据的审计以及数据质量管理。在数据存储和传输中，缺乏对数据文件完整性的检查。比如有些 FITS 文件的头信息不完整或者缺失、文件命名不规范。更严重的是传输时数据文件传输不完整，导致整个数据文件失效，造成不良的后果。
5. 缺乏数据的全局管理。比如针对国家天文台来说，缺乏对现有观测站点观测数据的纵览视图。作为管理层，如果需要了解整个数据的情况，数据中心将费很

大的力气进行统计才可以获知。这也是缺乏基础统计数据的问题。

这些问题随着中国国产数据规模越来越大，站点和望远镜越来越多，将会变得越来越突出。如果不解决这些问题，未来数据管理的难度将会更大。因此非常有必要对这些问题进行的调查与研究，为进一步建设中国自有的天文数据归档管理系统提供有益的支持。

### 1.2.2　国外天文数据以及研究现状

大部分天文数据的产生是由天文望远镜观测产生的，在国际天文界，每架望远镜或者每个天文台通常都有一套自己的天文数据归档策略和程序，并且基本都是围绕着天文数据预处理、处理以及数据获取展开的。

比较典型的如欧洲南方天文台（ESO，以下简称欧南台），它在智利拥有多个站点，多架望远镜。为此，欧南台构建了一套自己的数据归档系统[1]，它将数据归档视为数据流的一部分，包含有数据传输系统、数据仓库、数据库等。它在数据归档过程中还为数据预处理、数据处理等的需求准备了元数据库。在数据流的最后，自动构建其为最终用户使用的检索数据库。在欧南台数据归档的构建中，元数据库的建设是其重点建设的内容。以其 Paranal 天文台为例，2009 年望远镜每日产生的原始数据是 170GB，压缩后约 20GB，数据将分步传输至德国慕尼黑的欧南台总部。为此欧南台还建立了一套专门的数据传输系统来承担具体的工作。另外，欧南台也针对需求构建了数据管理的元数据系统[2]。

大型综合巡天望远镜（LSST）是正在建造的位于智利的一架直径 8.4m 的天文望远镜，它的 CCD 相机达到了 32 亿像素。它可以拍摄出 6 个波段的图像，预计 2020 年开光投入使用。以聚光能力和视野宽度来说，LSST 每周可以对整个南半球天空覆盖观测两次，每晚数据量将达 15TB。针对其数据管理访问，LSST[3] 也在逐渐构建满足其需求的分布式数据存储系统。

国际天文界自 20 世纪 70 年代以来，广泛使用了称作“普适图像传输系统（FITS）[4]”的文件格式进行数据存储。FITS 数据格式可以存储观测图像，也可以存储光谱以及星表等内容，是国际天文界广泛使用的数据格式。目前的最新正式版本是是 2010 年发布的 3.0 版本，2016 年发布了 4.0 的草案版本。

另外一个近十多年的重要发展是虚拟天文台。1999 年，美国科学院天文与天体物理发展规划委员会在名为“新千年的天文学和天体物理学”的未来十年发展规划中把建立（美国）国家虚拟天文台（NVO）作为最优先推荐的项目之一，首次提出了虚拟天文台的概念。此后在 2002 年，多个国家的研究机构合作组建了国际虚拟天文台联盟（IVOA）

[5]。中国虚拟天文台（China-VO）作为成员在 2003 年加入了该组织。

虚拟天文台致力于解决数据的互操作问题，在过去的十多年里，协商并发布了多个有关数据管理、数据互操作的协议，比如：天文互操作数据格式标准（VOTable)、锥形检索服务 (Cone Search)、表格访问协议 (TAP)、虚拟空间（VOSpace）协议等，更多的标准和协议可以查阅: http://ivoa.net/documents/。

## 1.3 研究目标和研究内容

### 1.3.1 研究目标

针对当前天文数据管理面临的困难与问题，本文的目标是构建一个满足中国天文数据管理与应用的全生命周期天文数据管理系统，重点解决的问题是**天文海量数据的存储管理问题**，并且面向天文海量数据的应用。全生命周期天文数据管理涵盖了数据从生产、存储、传输、分析、产品等一系列环节。核心则是数据的分布式云存储引擎。

本文研究的意义在于力图构建面向天文海量数据的分布式天文数据管理系统，可以有效的解决天文数据管理中的基本问题：数据流以及统计。

### 1.3.2 研究内容

为了达到上述研究目标，将重点进行下面的工作：

- 分布式存储文件系统的研究，主要解决底层数据存储以及高性能数据访问的问题。
  - 分布式文件系统尤其是基于对象存储的存储模型；
  - 面向天文应用的数据访问接口。
- 面向天文数据的数据传输算法和方案的实现，解决天文数据的传输高效、稳定等问题。
- 天文数据元数据的设计与实现，解决数据（文件型数据、结构化数据）的元数据管理问题，包括建立、入库、关联、检索等。

## 1.4 本文结构

本文共分为七章。

第一章为绪论，首先介绍了下论文的背景和相关的情况，引出了基本需求，说明了研究目标与内容。

第二章为相关技术与研究工作，介绍了与本文相关的天文数据管理、分布式数据管理等的相关知识、技术等研究工作。

第三章为系统需求与分析，详细介绍了面向天文海量数据管理的具体细致点需求以及分析，并且给出了需求用例。

第四章为系统总体设计，对系统进行了总体上的架构分析，并对各个模块、系统流程进行了设计和分析。

第五章为系统的详细设计与实现，针对系统的各个模块，尤其是核心的模块进行详细的设计。

第六章为系统测试与部署，针对系统的研发和部署，设计了一个基本的系统测试方案。针对不同的应用场景，详细描述了系统实施和部署的方式方法。

第七章对本文的工作进行了总结，并对以后的研究工作进行了展望。

# 第二章　研究现状与趋势

本章从两个方面对天文数据的存储与管理进行调研和分析，一是天文数据的管理现状，一是工业界已有的技术分析，并且为最终的技术分析和设计提供基础。

## 2.1　天文数据管理现状

天文数据和其它学科的数据相比有显著的特点。天文数据有自己的专用格式，主要是 FITS，天文数据还是世界上最开放的数据之一，任何人都可以免费获得。随着近些年来各个国家天文望远镜等项目的不断增多，天文数据总量也越来越大。天文数据按照观测任务分类，可以分为巡天观测数据和定点观测数据等；按照观测波段可以分为射电、红外、光学、紫外、$X$ 射线、$\gamma$ 等波段数据；按照数据性质可以分为图像数据、光谱数据、星表数据、时序数据、数值模拟数据等。[6]

### 2.1.1　天文数据简介

天文数据的来源有：天文望远镜的观测、天文计算数据模拟、望远镜运行等，根据天文学家的传统使用习惯，天文数据分为以下几类：

**图像**

天文望远镜拍摄的天文图像，这是最常规，也是最基本的一种数据。这类数据一般均为非结构化数据，是对天体的拍照，图像中的每一个像素与天空的一块区域对应，通过一定的加工处理流程（天文中称之为 Pipeline）可以计算出天体的一些性质，比如亮度、类型等。如图2，这是一张 BASS 巡天产生的星系图片。

**星表**

星表一般是对天体图像中每个天体的解读，通过 Pipeline，处理完图像，获得了图像中一系列天体的物理属性，这些天体物理属性组成了一个表，就是星表，星表中的一行代表一个天体，每一列代表一个物理属性，因此，星表是一个有个行列规则的表格，可以通过关系型数据库进行管理。

表1是一个简单的示例，是 LAMOST DR2 光谱星表的部分。

**光谱**

光谱是对天体更加细致的观测，常规的图像观测无法获得的一些细致精密的信息可以通过光谱进行获取的，比如，天体的元素丰度、视向速度等。在天文学领域，光谱还可以根据观测目标的不同，划分更细的类型。图3光谱示例就是 LAMOST 望远镜观测到

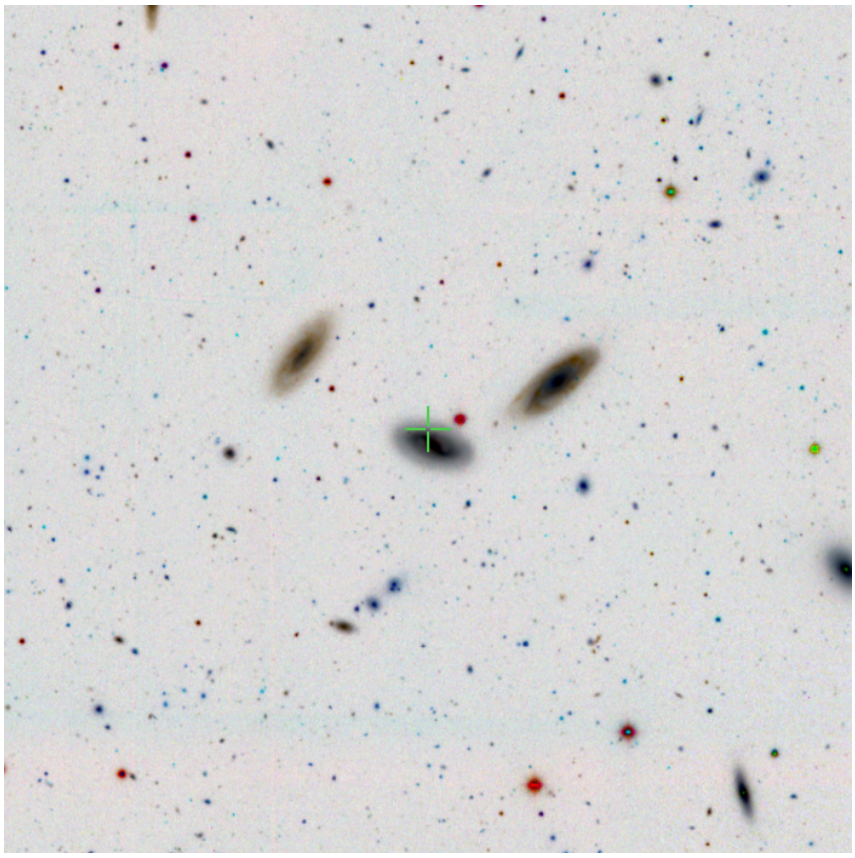

图 2　图像示例（BASS 巡天的一张星系图片）

表 1　星表示例（LAMOST DR2 的光谱星表部分）

| ID | RA | Dec | ··· | Mag |
|---|---|---|---|---|
| J220848.54-020324.3 | 332.2022740000 | -2.0567670000 | ··· | 18.78 |
| J220953.17-020506.0 | 332.4715760000 | -2.0850150000 | ··· | 20.91 |
| J220943.57-020344.5 | 332.4315490000 | -2.0623750000 | ··· | 23.43 |
| J221008.50-020659.1 | 332.5354560000 | -2.1164360000 | ··· | 18.87 |
| J220922.26-015509.0 | 332.3427820000 | -1.9191920000 | ··· | 20.31 |
| ··· | ··· | ··· | ··· | ··· |
| J221003.01-015029.0 | 332.5125580000 | -1.8414080000 | ··· | 22.89 |
| J220928.49-015720.7 | 332.3687450000 | -1.9557710000 | ··· | 18.25 |
| J220849.59-015207.1 | 332.2066650000 | -1.8686530000 | ··· | 18.64 |
| J220934.11-015156.0 | 332.3921300000 | -1.8655560000 | ··· | 21.11 |

的一条天体光谱。

**观测日志**

观测日志一般是望远镜的运行日志，记录的是望远镜运行时，在某时某刻观测了哪些目标，使用了哪些仪器，观测者是谁，等等。这些信息也可以使用关系型数据库进行

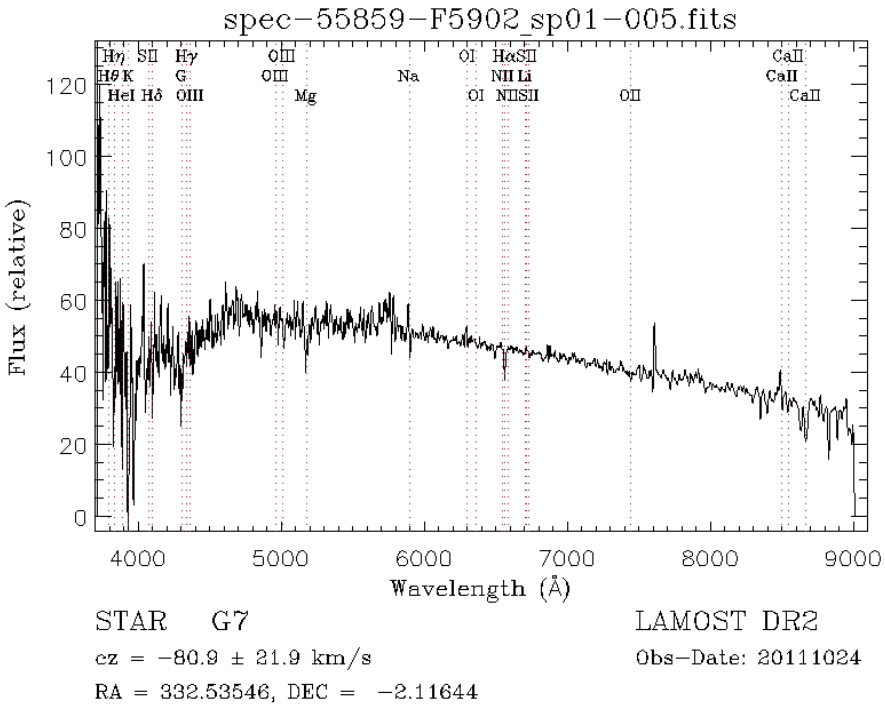


图 3　光谱示例（LAMOST DR2 的一条光谱）

存储和管理。

**环境信息**

环境信息是天文观测时的一些与观测相关的背景信息，比如气象信息、视宁度信息、天光亮度等。这些信息也可以使用数据库进行存储和管理。

### 2.1.2　天文数据标准

FITS（Flexible Image Transport System）[4] 是国际天文学联合会 (IAU）1982 年开始制定的标准数据格式，主要用于天文学家之间进行数据传输与交换。目前大部分主流的天文数据都为 FITS 格式。FITS 文件由文件头和数据体组成，在文件头中有对该文件的描述，如观测时间、历元、观测对象、观测人员、望远镜、曝光时间等信息，非常方便科研人员对其进行读取和使用。图4是一个典型的 FITS 文件头信息。

### 2.1.3　天文数据存储

传统上，天文数据的存储是存储中磁带、磁盘阵列、硬盘上的，根据需要进行获取和处理。但是随着天文海量数据时代的来临，对数据存储和管理的要求越来越高，比如以郭守敬望远镜（LAMOST）为例，每天晚上将产生 20-30GB 的原始数据，一万多个天体的光谱信息。截止到目前，已经获取七百多万条天体光谱。这给传统的 Linux 文件系

```
SIMPLE  =                    T /Primary Header created by MWRFITS v1.11b
BITPIX  =                  -32 /
NAXIS   =                    2 / Number of array dimensions
NAXIS1  =                 3909 /
NAXIS2  =                    5 /
EXTEND  =                    T /

COMMENT --------FILE INFORMATION
FILENAME= 'spec-56617-GAC122S02V2_sp16-239.fits' /
OBSID   =            180916239 / Unique number ID of this spectrum
AUTHOR  = 'LAMOST Pipeline'    / Who compiled the information
DATA_V  = 'LAMOST DR2'         / Data release version
EXTEN0  = 'Flux, Inverse, Wavelength, Andmask, Ormask' /
N_EXTEN =                    1 / The extension number
EXTNAME = 'Flux    '           / The extension name
ORIGIN  = 'NAOC-LAMOST'        / Organization responsible for creating this file
DATE    = '2015-12-17T10:09:26' / Time when this HDU is created (UTC)
COMMENT --------TELESCOPE PARAMETERS
TELESCOP= 'LAMOST  '           / GuoShouJing Telescope
LONGITUD=               117.58 / [deg] Longitude of site
LATITUDE=                40.39 / [deg] Latitude of site
FOCUS   =                19964 / [mm] Telescope focus
CAMPRO  = 'NEWCAM  '           / Camera program name
CAMVER  = 'v2.0    '           / Camera program version
COMMENT --------OBSERVATION PARAMETERS
DATE-OBS= '2013-11-20T20:22:35.42' / The observation median UTC
DATE-BEG= '2013-11-21T04:09:22.0' / The observation start local time
DATE-END= '2013-11-21T04:52:31.0' / The observation end local time
LMJD    =                56617 / Local Modified Julian Day
MJD     =                56616 / Modified Julian Day
LMJMLIST= '81528729-81528745-81528762' / Local Modified Julian Minute list
PLANID  = 'GAC122S02V2'        / Plan ID in use
RA      =           121.111080 / [deg] Right ascension of object
DEC     =            -0.860614 / [deg] Declination of object
RA_OBS  =           121.111080 / [deg] Right ascension during observing
DEC_OBS =            -0.860614 / [deg] Declination during observing

......

COMMENT --------SPECTRA ANALYSIS RESULTS
VERSPIPE= 'v2.7.5  '           / Version of Pipeline
CLASS   = 'STAR    '           / Class of object
SUBCLASS= 'F5      '           / Subclass of object
Z       =          0.00024400 / Redshift of object
Z_ERR   =          0.00007700 / Redshift error of object
ZFLAG   = 'LASP    '           / Which method computes the redshift
SNRU    =                21.90 / SNR of u filter
SNRG    =                97.12 / SNR of g filter
SNRR    =               117.59 / SNR of r filter
SNRI    =                71.19 / SNR of i filter
SNRZ    =                61.30 / SNR of z filter
END
```

图 4　FITS 文件头信息示例

统带来的存取的压力已经比较大，而当一些用户需要大批量下载数据的时候，性能更是迅速下降。因此迫切需要摆脱传统的思维进行数据的管理。

天文数据还有一个非常重要的特征是：天文数据分为原始数据（Raw Data）和产品

数据（Product Data），原始数据是望远镜直接观测的数据，一般包含大量的仪器信息，天文学家不容易使用，而天文学家最常用的是产品数据。原始数据存储的实际并非纯物理数据，而是包含了望远镜硬件信息、环境信息、天空背景等诸多信息，需要望远镜的运行团队进行处理，剥离附加的信息，还原为真实的物理参量，最终科学家得到的是产品数据。

原始数据可以理解为冷数据，实际上一般只在望远镜的数据中心存储，而且使用的次数很少，但需要长期保存。而产品数据则不同，它是热数据，天文学家、公众都可以通过网络在线下载使用，使用的频度比较高。

不同于商业和网络上的使用，比如像 Facebook、新浪微博、淘宝等的海量文件管理中，常见的场景是大量写入、大量读取、大量的修改删除。而在天文数据中是非常简单的：一次性写入、大量读取、极少修改删除。这样，我们的系统将可以针对这个场景做更多的优化，摒除大量不用的功能。

## 2.2 虚拟天文台与大数据

### 2.2.1 虚拟天文台

虚拟天文台技术是近十多年在国际天文界迅速发展的一系列技术的统称。虚拟天文台（The Virtual Observatory）是利用先进的信息技术将各种天文研究资源以统一的服务模式无缝透明地汇集到一起，形成一个统一的数据密集型的网络化天文研究与科普教育平台[7][8][9]。

国际虚拟天文台联盟（The International Virtual Observatory Alliance，IVOA）是在 2002 年 6 月，由多个国家的研究机构合作组建[5]。中国虚拟天文台（China-VO）作为成员在 2003 年加入了该组织。

国际虚拟天文台联盟目前由包括中国虚拟天文台在内的 21 个成员组成，如图5。各成员联合成立了国际虚拟天文台联盟执行委员会，下设秘书和技术协调组，并且按照侧重点的不同又细分为工作组和兴趣组。

中国虚拟天文台[1]旨在完成国际虚拟天文台联盟宏伟构想中的中国部分, 正在以国内核心天文观测设备的时间申请、审批，数据汇交、共享、使用，课题设计、开展为线索, 融合天文观测和科研活动所需的科学数据、科技文献、高性能计算、软件和实用工具等资源, 打造一个物理上分散、逻辑上统一的网络化科学研究平台; 基于虚拟天文台技术和云计算技术实现一个全生命周期数据管理与开放共享平台[9]。中国虚拟天文台是一个数据驱动的科研信息化环境, 基于标准、完整、有质量保障的元数据和科学数据系统,

[1]http://www.china-vo.org

通过具备互操作能力的软件、工具和服务, 为天文学家等科学用户打造一个信息化科研新模式。同时, 这是一个开放的平台, 通过标准的接口和协议与国际上的资源和服务实现无缝融合。


图 5　国际虚拟天文台联盟成员

### 2.2.2　大数据时代的天文学研究

天文信息学[10][11][12] 是研究天文信息的获取、处理、存储、传输、分析、挖掘和解释等方面的学科。它是天文学、天体物理学、计算机科学、工程学和信息学相结合的一门新型学科，应用天文学、天体物理学、计算机科学、工程学和信息技术揭示大量复杂的天文数据所赋有的宇宙和天体的奥秘, 天文信息学旨在为天文学和信息技术以及计算机应用科学搭建桥梁, 以基于 VO 框架建立起来的数据网格为基础, 为数据密集型天文学的研究者们提供一个更广阔的社区。[13]

大数据的时代已经来临，首先是数据规模，其次天文数据以其获得方式的不同，主要分为观测数据、数值模拟数据；以其存在方式的不同，可以分为图像、星表、光谱、时序数据、网页数据；以其结构的不同，可以分为结构化数据、半结构化数据、非结构化数据。天文数据的特点包括：空间性、高维性、海量性、多模式、多尺度、多分辨率、时序性、缺值性、带误差、异构性、分布性、开放性等。天文数据的复杂性体现在：高光谱、非线性、非高斯噪声、非线性系统影响、形状和大小的变化、密度的变化、近似性、局部维数不同等。这些特点和复杂性都对天文数据的存储、传输、处理、分析和挖掘提出了严峻的挑战。[10][14]

## 2.3 分布式存储

“分布式存储系统是大量普通 PC 服务器通过 Internet 互联，对外作为一个整体提供存储服务”[15][16]。

分布式文件系统具有三大主要特征：

- 透明性：用户访问系统，不管从哪里登陆，用户体验和本地文件系统应该是一样的，用户不用关心底层是如果构造的，也就是说，底层系统的复杂性对用户是隐藏的；
- 扩展性：分布式系统可以进行快速扩展以达到扩容的目的，而且可以实现在线弹性扩展；
- 容错性：发生问题故障时，系统不应该停止，应该有一定段容错机制，保证数据的一致性以及服务的可靠性。

下面是一些工业界在解决海量数据存储时采用的一些技术方法和方案，并进行了一个简单的评述，以吸取其精华，为构筑自己的系统做调研。

### 2.3.1 分布式文件系统与对象存储

为解决海量文件，尤其是海量小文件（比如：头像、照片等）等存储管理问题，Google、Facebook、阿里巴巴、百度、奇虎 360 等纷纷提出了自己的解决方案，有的甚至在得到了大规模使用后进行了开源。

Facebook 需要解决的问题海量图片文件的存储瓶颈，图片的特征是海量、尺寸小、写多、读多。在当时的情况下，每周 Facebook 都有约 60TB 的图片上传，依赖于于传统的方法很难解决这个问题。传统的文件系统中，以 Linux 文件系统为例，文件的元数据存储在 inode 中，当文件规模达到一定等程度后，元数据的查询性能急剧下降。

Facebook 的 Haystack 方案[17][18] 是一个典型的分布式存储架构，技术的核心是将海量等小尺寸文件堆砌中一个大文件中，减少系统级的 inode 元数据访问压力，而对每一个真实文件的访问则是采用接口实现，接口会在索引文件或者数据库中获取真实文件在大文件中的偏移量以及数据体大小。

Haystack 的架构主要由三部分组成：Haystack Directory，Haystack Store 和 Haystack Cache。Haystack Directory：目录服务，是数据接口的第一步，提供了对数据访问的目录管理功能，用户请求后，会给用户返回 cache 和 store 的信息。同时还可以提供高可用性、负载均衡的一些功能，比如告诉用户使用哪个节点，这个对于分布式的服务非常有用。而在写入时，目录服务还需要为写请求分配图片的 ID。

在一个超级块里，有若干个 Needle，Needle 对应的是逻辑文件。每个 Needle 由若

干字段组成，比如有编号，大小，内容，检验码等。参考这个的设计思路，本本将定制自己的 Needle 格式，以满足天文数据的要求，另外，Haystack 的目标主要是图片存储服务，因此其字段多考虑了图片的信息，比如 Alternate Key 字段。

超级块也被称作物理卷轴（Physical Volume），物理卷轴的大小一般是 100GB，这样一个 10TB 的存储节点只需要存储 100 物理卷轴即可，多个物理卷轴可以组成一个逻辑卷轴，这个对于天文数据来说，一个逻辑卷轴可以对应一个数据集，天文数据的特点是稳定性很好，极少需要更新和删除。可以在生产一个物理卷轴后后台自动进行副本的复制，提高可用性。

每个物理卷轴还拥有一个索引文件，保存每个 needle 的元信息，比如 needle 的编号、在文件里的偏移量等，为快速访问数据提供支持。写操作是首先写入物理卷轴文件，然后更新索引文件，因此是异步的。另外物理卷轴文件和索引文件都是典型“追加”模型，维护也比较方便。

Haystack Cache 是缓存服务，主要是将常用的数据放入内存中，加快数据访问速度。

Haystack Store 是核心的存储服务，将大批量的逻辑文件合并成一个物理文件，这样可以快速地降低 Linux 文件系统 inode 的数量，提高读写效率，同时也可以高效地进行数据的多副本复制。

目前基于 Haystack 的思想已经有了若干开源的实现，比如哔哩哔哩公司毛剑开发的 bfs[19][2]和 SeaweedFS[3]，这两者均是基于 Go 语言开发，并根据各自的业务需求，进行了定制。

Google 在更早之前的 2003 年提出了它的分布式文件系统解决方案[20]，提供的也是海量数据的分布式服务，与 Haystack Store 类似，GFS 有 Chunk Server，与 Haystack Directory 对应的则是 GFS Master。

GFS 由一个单 master 和多个 Chunk server 组成。GFS Chunk 的大小与 Haystack 的不同，每个块的大小是 64MB，并且拥有唯一的 ID，块和块副本都是以文件的形式在服务器中存储。

GFS 的 Master 服务也是一个中枢的服务，客户端需要从 master 了解文件具体分布在哪些 chunk 块上。因此 master 有可能成为瓶颈。Master 存储三种类型的元数据：文件和块的命名空间、文件到块 k 的映射、每个块副本的位置。所有的元数据都保存在 master 的内存中。

GFS 提供了常见的文件系统的接口，比如目录分层管理等。但不同于传统文件系统，GFS 不能够列出目录下所有文件的目录数据结构。也不支持同一文件或者目录的别

[2]https://github.com/Terry-Mao/bfs

[3]https://github.com/chrislusf/seaweedfs

名（例如，Unix 语境中的硬链接或者符号链接)。GFS 将其名称空间逻辑上表现为全路径作为一个类似主键的方式存在，以方便查找。

目前广泛得到使用的 Hadoop Distribute File System（HDFS）就是参考 GFS 的开源实现。并且重点面向海量数据的读写应用。

阿里巴巴为了解决其在淘宝系统上的海量数据管理问题，开发出了 TFS（Taobao FileSystem）[4]，是一个高可扩展、高可用、高性能、面向互联网服务的分布式文件系统。主要针对海量的非结构化数据，它构筑在普通的 Linux 机器集群上，可为外部提供高可靠和高并发的存储访问。TFS 为淘宝提供海量小文件存储，通常文件大小不超过 1M，满足了淘宝对小文件存储的需求，被广泛地应用在淘宝各项应用中。它采用了 HA 架构和平滑扩容，保证了整个文件系统的可用性和扩展性。同时扁平化的数据组织结构，可将文件名映射到文件的物理地址，简化了文件的访问流程。其设计思路与 Google GFS 类似，一定程度上为 TFS 提供了良好的读写性能。

百度的分布式文件系统解决方案是百度文件系统（BFS）[5]，旨在支持实时应用程序。与许多其他分布式文件系统一样，BFS 容错能力非常强。但与其他方面不同，BFS 提供低读取/写入延迟，同时保持高吞吐速率。

奇虎 360 的分布式文件系统解决方案是 huststore[6]，其功能更为丰富，是一个高性能的分布式存储服务，不但提供了 10 万 QPS 级别的 KV 存储能力，还提供了 hash、set、sort set 等一系列数据结构的支持，并且支持二进制的 KV 存储，可以替代 Redis 相关的功能。此外，huststore 还结合特有的 HA 模块实现了分布式消息队列的功能，包括消息的流式推送，以及消息的发布-订阅等功能，可以替代 RabbitMQ 或者 Gearman 相关的功能。

纵观几个分布式文件系统，大多是分布式对象存储的设计，而且都有相似的设计思路，主要包含一个底层文件存储系统（Haystack Store，Chunk Server 等）以及一个目录(元数据、中枢）服务（GFS Master，Haystack Directory 等)。

### 2.3.2 重构分布式文件系统

基于之前的分析，本文试图构建的技术方案核心将是一个对象存储系统、一个分布式文件系统。但为什么没有选择这些开源实现，而选择重新构造一个新的分布式文件系统，主要基于以下几个方面的考虑：

1. 现有解决方案过于重：根据天文数据存储管理的特点，目前的主要开源系统稍

[4]http://tfs.taobao.org/

[5]https://github.com/baidu/bfs

[6]https://github.com/Qihoo360/huststore

显复杂，功能过于庞杂。比如在天文数据管理中，读写的频度和量级远远达不到上述工业界解决方案的需求。天文数据基本都是一次性写入，不需要多次写入。因此，在构造自己系统的时候，可以大幅度的精简结构；

2. 真正面向天文数据管理的分布式文件系统还没有，而且由于应用场景的不同，桌面端或者一个简化的版本还是有很大用武之地的；
3. 真正面向天文海量数据应用的文件系统还没有，也就是面向天文海量数据处理的高性能接口还没有，天文海量数据的处理需要有其特殊的方法与工具。

本文的方案也借鉴了 Facebook 的 Haystack 方案以及 Google GFS 方案的一些基本思想，比如小文件合并成大文件存储、数据块索引等。但考虑到天文学的实际情况，做了一些的优化和创新。

另外，我们选择构建的系统是一个分布式对象存储系统，并没有选择构建块存储系统，其中一个主要原因是业务的需求。天文学的绝大多数应用目前来说，实时性要求没有那么高。针对实时性比价高的需求，未来可以考虑块存储模型，以满足这些使用场景，在本文中，将不予以研究。而且对于分布式对象存储来说，我们的目标是可以进行桌面部署简化版的，这样的情况下，对象存储更加适合些。

## 2.4　小结

本章详细介绍了相关的天文数据管理、分布式数据管理等方面的知识、技术以及研究工作。最后确定了重新构造一个面向天文数据管理等一个分布式文件系统，系统模型则是参考 Facebook 的 Haystack、Google 的 GFS 等实现模型。

# 第三章　系统需求与分析

## 3.1　需求综述

本文主要的研究对象是天文海量数据，其目标是怎么管理和维护好这些数据，因此关键技术是为此构造一个的分布式文件管理系统、一个面向天文海量数据管理的基础数据管理设施，重点在于构建底层的存储引擎，然后在此基础上可以构建出数据服务的具体应用。

2015 年，中国科学院正式成立了“天文大科学研究中心”[1]，中心由中国科学院国家天文台（含总部、云南天文台、南京天文光学技术研究所、新疆天文台、长春人造卫星观测站)、中国科学院紫金山天文台和中国科学院上海天文台共同建设。旨在汇聚中科院和高校优势力量，在重大科学问题牵引下，开放运行和科学发展天文大科学装置及重要天文观测设备，组织大团队科学攻关和核心技术研发，产出重大原创性成果，开展高水平国际合作，为前沿科学研究、科教融合和科学传播提供一流的开放共享服务，成为集世界先进观测装置群、一流天文技术研发平台、高端天文科技人才队伍于一体的世界知名天文大科学研究中心。

这天文大科学研究中心的定位中，明确要求了天文科学数据的管理工作，以中国虚拟天文台为基础打造“**天文观测服务平台**”，将天文数据的全生命周期管理纳入到其中。

在天文数据的管理中，参与数据管理和使用的角色按照参与时间的顺序主要分为了以下几类：

1. **数据生产者**，主要是望远镜的运维团队，也是数据生产的源头。

2. **数据拥有者**，根据天文数据的一般应用习惯，望远镜的观测来源于观测计划，观测计划来源于天文学家的申请，通过评审后通过的观测计划才可以被执行，该天文学家拥有对数据的优先使用权。按照天文学界的惯例，数据一般有 1-2 年的保护期，在保护期内，该天文学家可视为数据的拥有者，保护期过后，数据将按数据政策对外免费开放，此时数据的拥有者将是望远镜的运维团队。

3. **数据使用者**，数据的最终用户，包括科学家、感兴趣的公众等。

4. **数据客户端**，以上的参与者主要都是人员，数据客户端则是满足了虚拟天文台相关协议的程序或者软件，它们通过一定的接口来实现对数据的访问。

5. **天文管理机构**，主要是对望远镜运维的资助者和管理的高层管理者，比如天文台管理层、天文大科学研究中心管理层、国家科学大装置办公室等。他们关注的是望远

[1]http://www.nao.cas.cn/xwzx/zhxw/201509/t20150917_4426340.html

镜的观测效率、数据的使用效率以及科学产出的指标等，更多的是统计数据。

以下将针对数据流以及在数据流上的参与者详细描述其对数据管理等需求，并以此指导最终系统的设计与实现。

## 3.2　功能需求

天文数据的全生命周期，如图6，天文数据的全生命周期[21][22][23] 指的是从望远镜的观测时间申请开始：

1. 天文学家根据望远镜的要求提交观测申请；
2. 一个资深天文学家组成的委员会审查这些观测申请，并对符合要求的申请进行批准；
3. 获得批准的申请将根据一定的算法被分配在某一天的某个时段进行观测；
4. 望远镜根据观测计划进行观测，观测结果和观测相关的日志等信息需要同步进行存储和管理；
5. 观测结束后，观测数据进行归档以及交给申请者进行研究；
6. 如果执行的是大型的观测计划，数据将会由一个数据处理分析团队进行数据处理，并根据该望远镜的数据政策进行发布；
7. 发布的数据是天文产品数据，天文学家、公众可以登录数据发布网站检索下载数据进行分析或查看；最后是产生科学产出，比如科学论文、科普文章、科普视频等。

在天文数据的全生命周期中，数据的管理是一个主线，每个环节都需要对数据进行管理，这也构成了整个系统需要解决的核心业务问题。

### 3.2.1　数据生产者

数据生产者是望远镜的运维团队，也是数据生产的源头，对于运维团队来说，其需求是：

- 望远镜时间申请系统，可以处理天文学家在线申请望远镜观测申请，可以邀请专家进行在线评审，并最终产生观测计划；
- 望远镜运维管理系统，负责记录望远镜观测的运行情况，比如观测目标、观测计划、观察者、观测类型、文件管理情况等；
- 数据管理平台，管理历史和现在的望远镜观测数据，并可以对数据进行检索、下载、分发等。

这几个需求可以通过观测计划的线索串联起来，形成一个完整的望远镜管理平台。

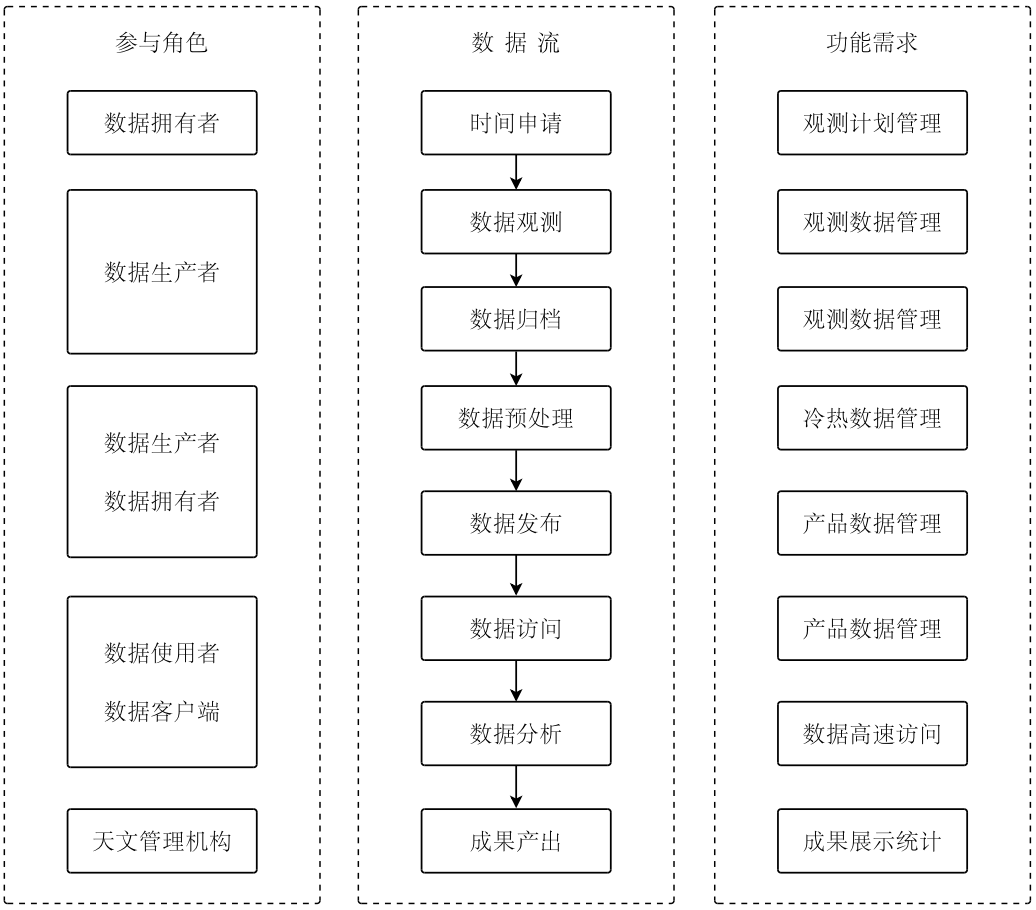


**图 6　天文数据的全生命周期**

但对于本文的目标来说，重点在数据的存储基础设施上，因此对与存储无关的功能暂时不予考虑。

### 3.2.2　数据拥有者

数据拥有者一般也是数据的科学用户，他比较关心的是望远镜的时间申请，在申请到的时间里观测的数据如何获取等。

### 3.2.3　数据使用者

数据使用者主要包括科学家和公众，都是希望通过一个界面（一般都是 Web 界面）进行数据的检索和下载，再深入些就是希望一些数据可以在线进行简单的处理。

根据目前大多数天文学家的科研习惯，科学数据还是希望可以下载到自己到科研计算机上进行处理，而现在海量的天文数据下载将是一个噩梦，在云端进行数据处理是未来的一个大趋势。科学用户还希望可以有一个稳定安全、高效的天文数据下载工具。

但随着天文海量数据时代的来临，数据将主要存储在云端，计算资源也在云端，让计算靠近数据是未来的大趋势。

### 3.2.4　数据客户端

数据客户端主要指的是程序和软件。基于虚拟天文台联盟制定的相关数据互操作协议，目前国际上已经有若干软件和程序对数据中心的数据进行互操作访问。

- `Aladin`[2] 是法国斯特拉斯堡数据中心开发的一个星表星图软件，背后依靠的就是其数据中心（以及全球多个镜像数据中心）的海量数据，并且通过标准的虚拟天文台协议进行检索、下载。如图7，软件的界面；
- `TOPCAT` [3] 是一个交互式天文图形查看器和表格数据编辑器[24]。如图8，目的是提供天文学家分析处理星表和其他表格数据的工具，它可以处理许多不同的天文数据格式（包括 FITS、VOTable、甚至是 CDF）。也提供了访问世界上大量天文数据中心的接口，这些接口大多数都是满足虚拟天文台的访问协议的。
- `AstroPy`[4] 是一个天文数据处理的 Python 程序库[25]，提供了大量的天文数据结构和算法，也提供了多个虚拟天文台访问协议接口库，方便用户通过自己编写程序访问满足虚拟天文台协议的数据中心数据。AstroPy 不仅仅是一个程序库，而且在其基础上制定了一套基于 AstroPy 开发程序包的规范，诸多的遵循其规范的程序也已经构建起来[5]。

### 3.2.5　天文管理机构

对于天文相关管理机构，比如天文台科研管理部门、台站、国家大科学装置办公室等，需要定期获得各个望远镜的运行状态、运行性能、科学产出等统计数据。其中一些基础的数据需要数据管理系统来提供，比如：

- 按年月日统计的望远镜观测数据量；
- 各个望远镜数据的使用量，都哪些人使用？
- 数据汇交情况；
- 各个望远镜的数据量情况；
- 其他感兴趣的信息。

## 3.3　虚拟天文台支持

虚拟天文台（Virtual Observatory）是利用先进的信息技术将各种天文研究资源以统一的服务模式无缝透明地汇集到一起，形成一个统一的数据密集型的网络化天文研究与

[2] http://aladin.u-strasbg.fr/

[3] http://www.star.bris.ac.uk/~mbt/topcat/

[4] http://www.astropy.org/

[5] http://www.astropy.org/affiliated/index.html

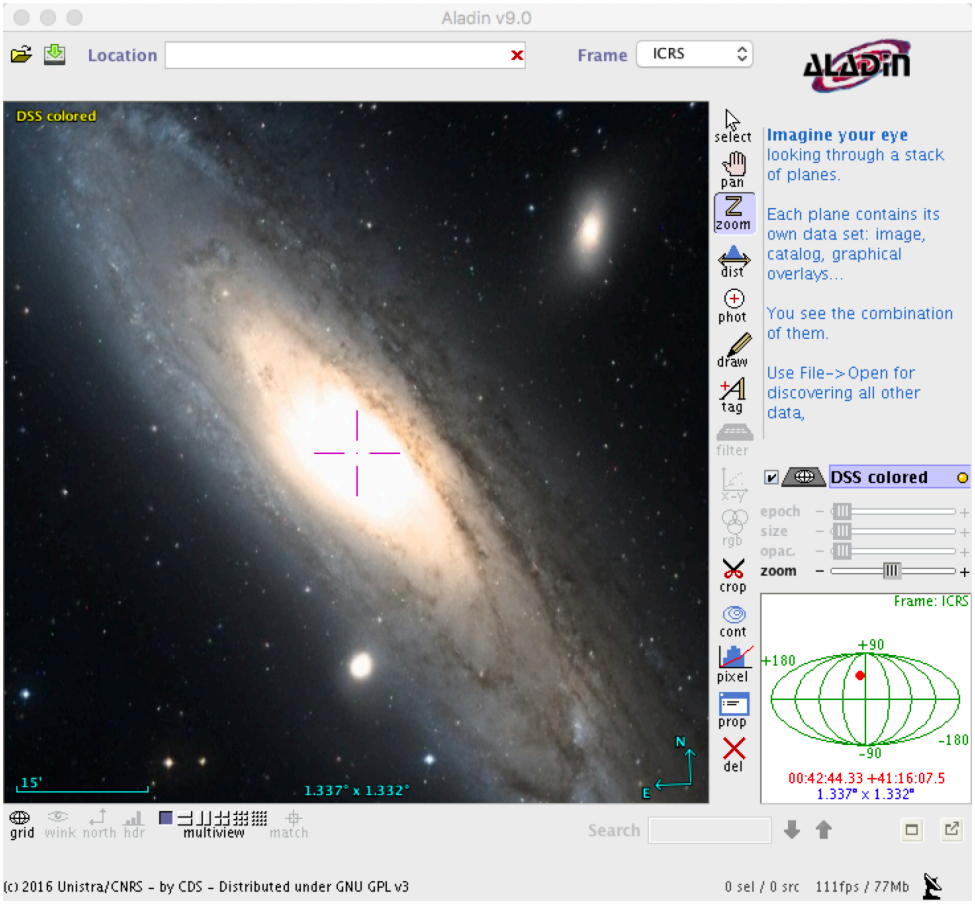


图 7　天文数据相关软件-Aladin

科普教育平台[7][8][9]。

虚拟天文台致力于解决数据的互操作问题，在过去的十多年里，协商并发布了多个有关数据管理、数据互操作的协议，比如：天文互操作数据格式标准（VOTable）、锥形检索服务 (Cone Search)、表格访问协议 (TAP) 等等，更多的标准和协议可以查阅: http://ivoa.net/documents/。

作为一个底层的系统，以及一个管理数据集的系统，需要并关注遵循的标准有：数据集元数据标准和 VOSpace 规范标准。但是虚拟天文台的这些协议更多关注的是数据的互操作，也就是更高层面对数据的描述，而本系统更偏重于底层，因此在设计中遵循的原则是实现相关协议的一个子集，为未来在上层封装这些服务提供基础。

### 3.3.1　数据集元数据

数据集元数据描述的是望远镜观测数据或者产品数据的元信息，国际虚拟天文台联盟制定了“IVOA 标识符规范”[26] 和“IVOA 数据服务协议”[27]。

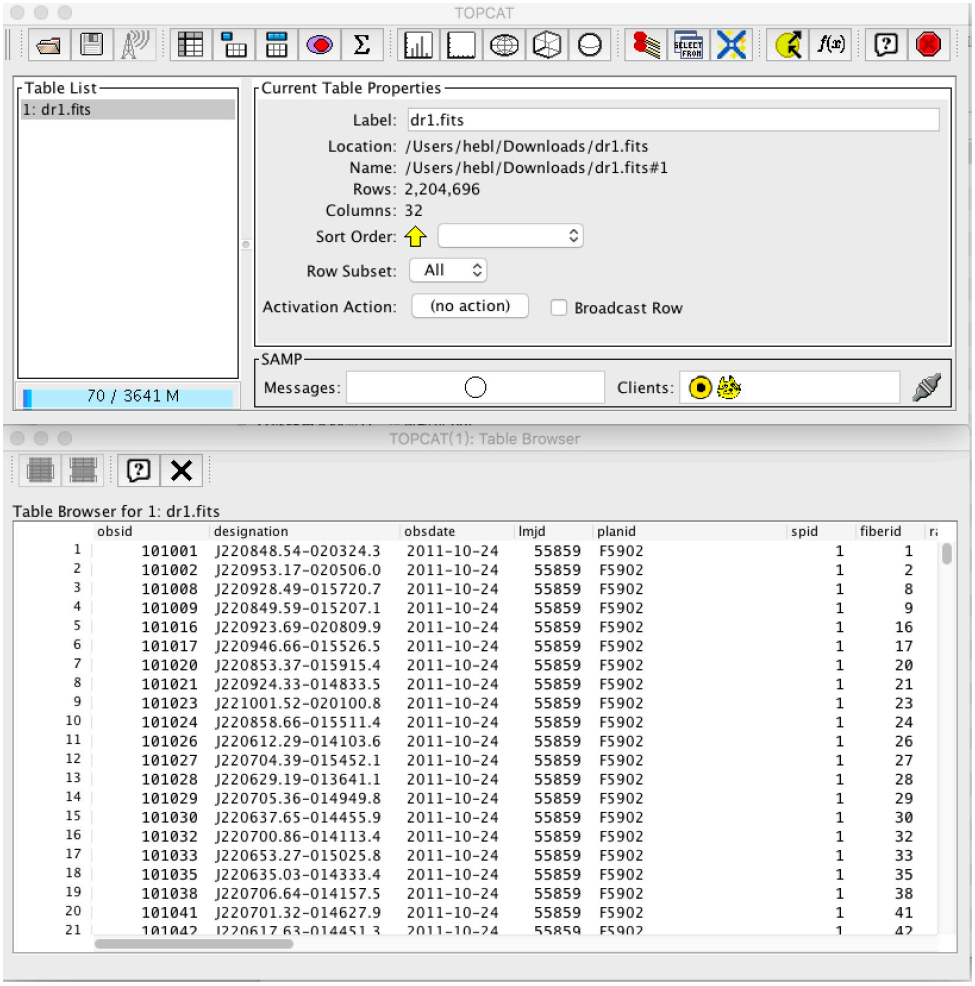


图 8　天文数据相关软件-Topcat

### 3.3.2　虚拟空间

虚拟空间（VOSpace）[28] 是基于分布式存储的虚拟天文台接口协议。以 RESTful 模式的 Web Service 服务实现对数据文件资源的标准访问。

VOSpace 服务是分布式存储网络的接入点。通过这个接入点，客户端可以：

- 新增和删除树形数据结构中的数据对象。树形数据结构类似于目录树，数据对象则可理解为数据文件，这也是典型的对象存储方式；
- 管理数据对象的元数据；
- 通过资源 URL 可以实现对文件对访问。

VOSpace 对数据的存储和传输没有说明，其重点在于数据的访问模式，因此在一个已有的文件系统中经过包装即可实现一个 VOSpace 服务。暴露给用户的是一个类似文件系统的接口。

## 3.4 应用场景分析

以下是两个典型应用场景的描述:

### 3.4.1 望远镜数据管理

1. 望远镜观测天体数据：参数数据、原始数据、环境数据、仪器数据等;
2. 望远镜管理人员将数据推送至数据节点的方式;
3. 数据节点进行数据标准化，元数据入库等操作;
4. 数据归档至国家天文台数据中心;
5. 国家天文台数据中心根据规则进行数据的水平分割，多副本，分布式存储等;
6. 完成归档;
7. 数据异地灾备（科技云、商业云等)
8. 完成备份。

### 3.4.2 科学家使用数据

1. 科学家从国家天文台数据中心通过界面、API 等方式获取数据;
2. 数据处理;
3. 处理结果通过界面、API 等归档至国家天文台数据中心，数据自动进行分布式处理和存储;
4. 数据分享（个人数据云)。

## 3.5 需求用例

根据前面的需求分析，可以提炼出一些的典型的用例。

### 3.5.1 望远镜数据节点

如图9，望远镜节点对数据管理等需求有:

- 数据管理操作：入库、查看、下载、统计等;
- 数据运维：望远镜观测站点与数据中心的交互，数据汇交，数据备份等。

在这个应用中，数据流是：望远镜观测 → 数据入库 → 数据统计 → 数据汇交。

### 3.5.2 主数据中心节点

如图10，在数据中心节点，有两方面的主要应用，一个是面向底层的服务管理，另一个是面向用户的数据接口。服务管理负责管理底层的数据、数据库、日志、副本、健

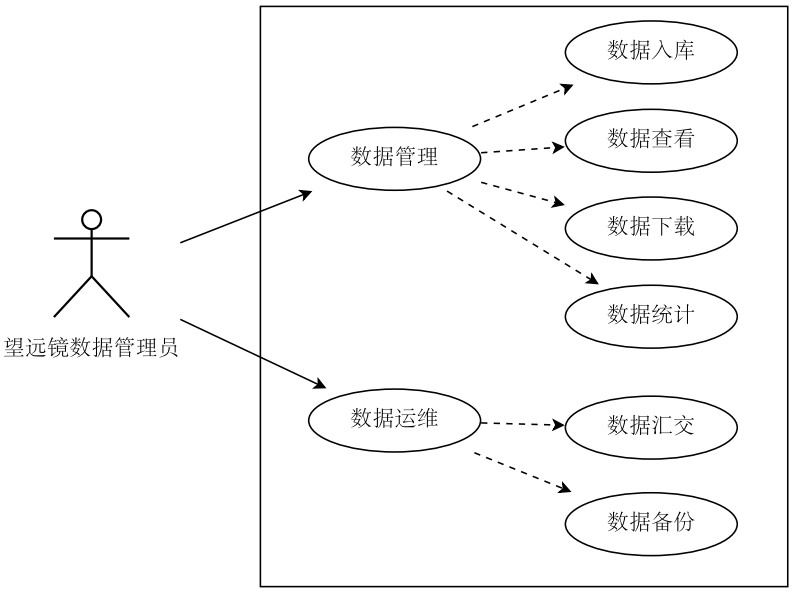


图 9　望远镜数据节点用例图

康检测以及子节点。通过数据接口则可以实现：数据上传、查看、下载、统计等。

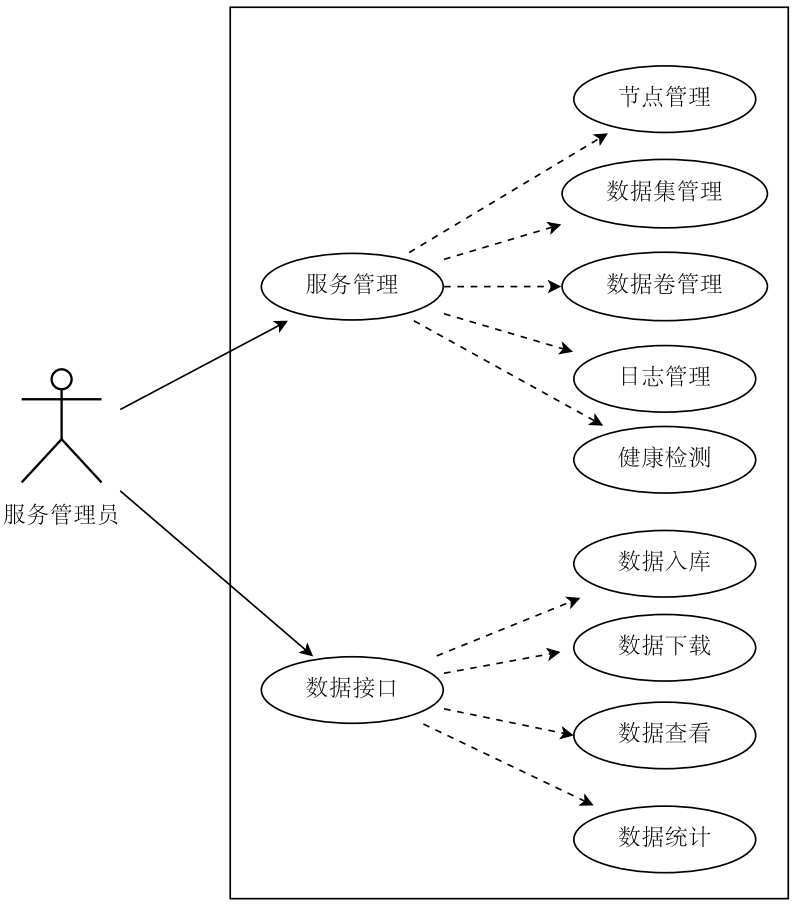


图 10　主中心数据节点用例图

### 3.5.3 普通数据用户

普通数据用户是数据的主要使用者，一般只拥有对数据的只读权限。如图11，其主要的用例就是查看、下载和统计数据。

数据用户也可以是客户端程序，根据访问协议，用户可以编写程序实现对数据的访问。

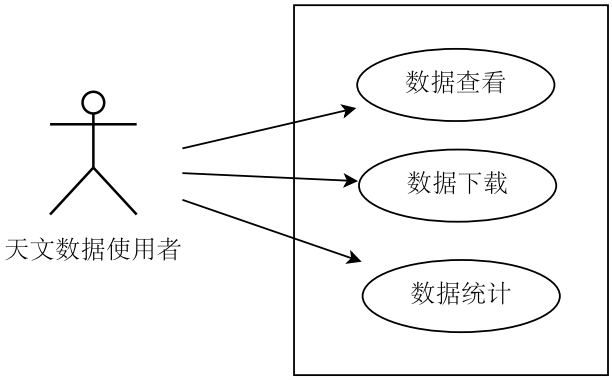


图 11　数据用户用例图

## 3.6 小结

本章详细介绍了系统的具体需求以及应用场景，包括功能需求，虚拟天文台支持，并且给出了两个典型的数据应用场景，并最终提炼出几个典型的用例。根据这些需求和主要应用场景，下面的章节将设计规划系统的架构，系统的流程。

# 第四章　系统总体设计

系统的整体目标是分布式文件系统，核心是底层的存储引擎以及易于实施的多种部署方式，因此从整个系统架构和整体设计上遵循的原则是：“层次清晰，逻辑串接”。

## 4.1　系统架构

整个系统是一个分布式文件系统，组网后的拓扑结构如图12，系统有不同的子节点，子节点可以是分布式模式，也可以是简化的小系统部署，甚至只是一个纯粹的客户端，而将管理功能委托给主节点。

主节点是一个分布式架构的系统节点，拥有全部的功能。

### 4.1.1　分布式天文数据存储网络

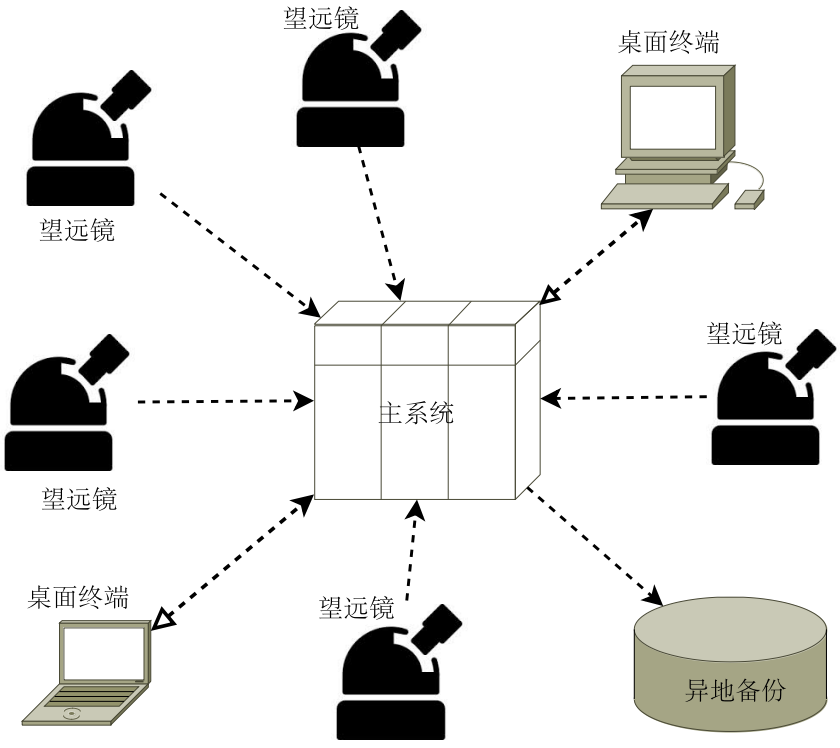


图 12　分布式天文数据存储网络

### 4.1.2　主系统架构

主系统的整体架构如图13，这是一个分布式的架构，主要分为三层：

1. GUI **和客户端**是系统直接针对用户的服务，包括一个用户（管理）界面，针对

用户或者客户端的接口，用户可以上传、下载和管理数据。

2. API 是对所有服务的封装，也是对内和对外的一个有效隔离。
3. **分布式系统**是系统的核心组件，包括有

**目录、调度服务** 提供给用户路由服务，分发用户的上传下载请求至底层的分布式存储服务，并且与中心数据库交互，负责数据集的注册、ID 的命名以及统一管理等。

**中心数据库** 这是一个高可用的关系型数据库，存储和缓存节点、数据集、数据卷以及数据文件的元信息、这些元信息在数据归档过程中通过逐级汇交等方式汇总在中心数据库中。

**运维** 包括监控、归档、副本同步等服务。

**分布式存储服务** 底层存储的核心分布式存储服务，由若干个节点构成，每个节点如图17，数据集默认为三副本，并且存储在不同的节点中。

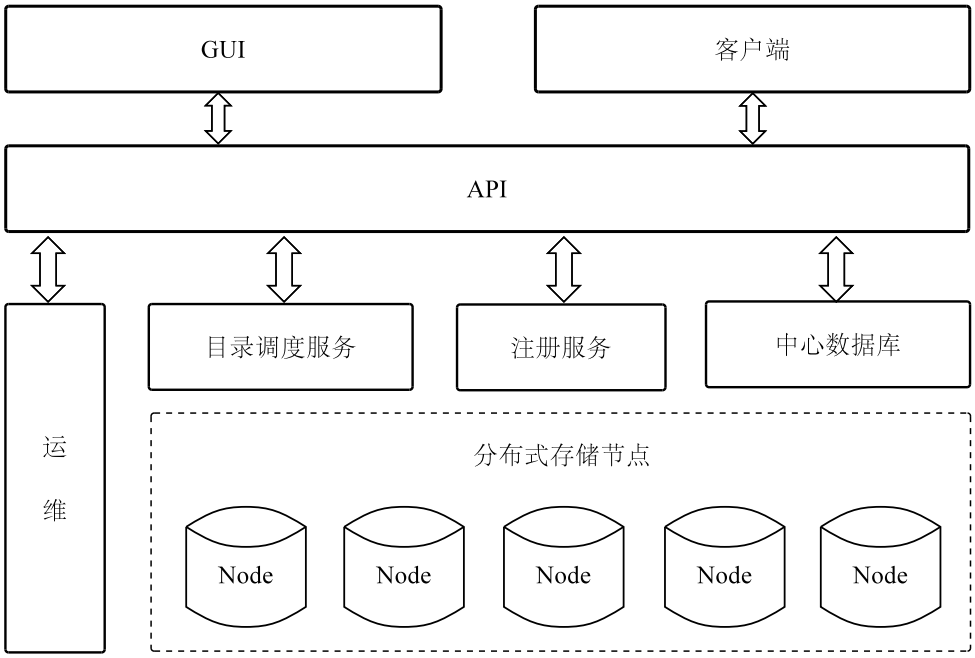


图 13　系统架构

### 4.1.3　小系统（单机版）架构

根据天文数据管理等需求，将在多个望远镜节点部署服务，因此，可以对系统架构进行简化，如图14，形成一个单机版程序。单机版程序即可以在单个服务器部署，也可以在桌面端部署。而桌面端，实际是一个个人管理天文数据的数据管理系统。

通过组合一个主系统以及多个小系统（单机版），如图12，就可以形成以前覆盖多个望远镜、终端，多节点的天文数据存储网络。

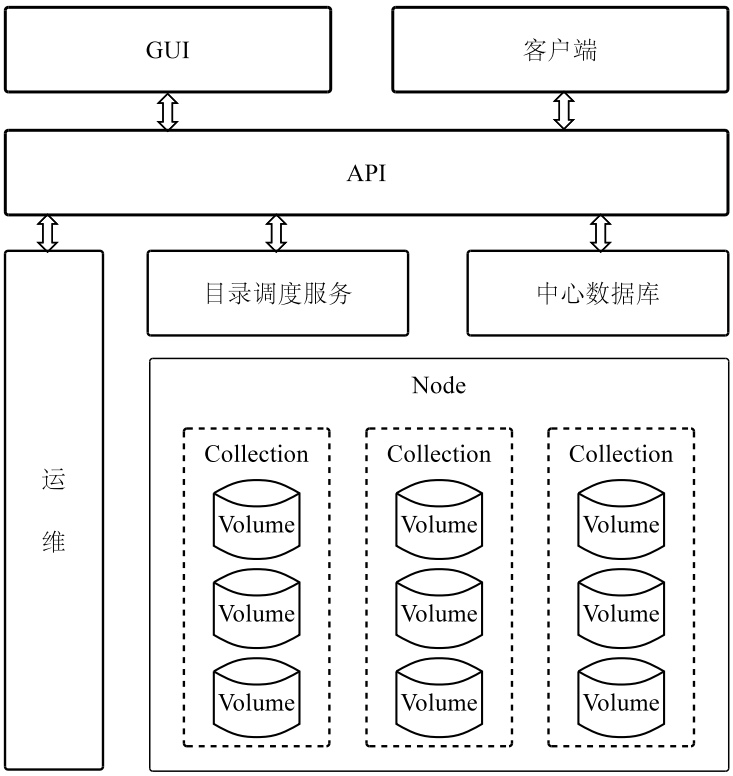


图 14　小系统（单机版）架构

## 4.2　系统功能

### 4.2.1　系统功能流程

根据系统总体的架构，如图15所示，可以将系统分为对外接口、目录调度、分布式存储、中心数据库、运维几大模块。模块之间通过接口进行解耦，并且可以通过不同的配置组合形成不同层次的系统版本。

如图15，各个模块之间有一定的调用关系，详细设计可以参考20和图21。

### 4.2.2　系统功能模块结构

根据系统总体架构，如图16所示，可以将系统分为五个主要的模块：对外接口、目录调度、分布式存储、中心数据库、运维，这些模块每个又可以分为若干子模块。

下面对各个模块进行分析和设计。

#### 对外接口

对外接口服务对外屏蔽了分布式系统内部的细节，用户或者前端只需要通过面向资源服务的 RESTful API 就可以使用本系统服务。

对外接口可以分为以下几类服务：

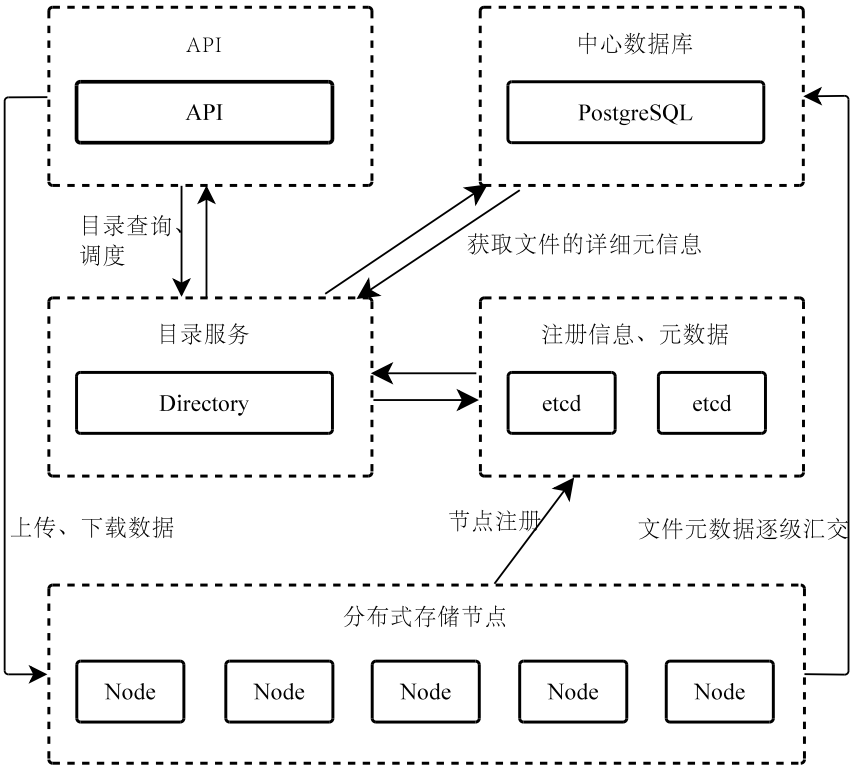


图 15　系统功能流程

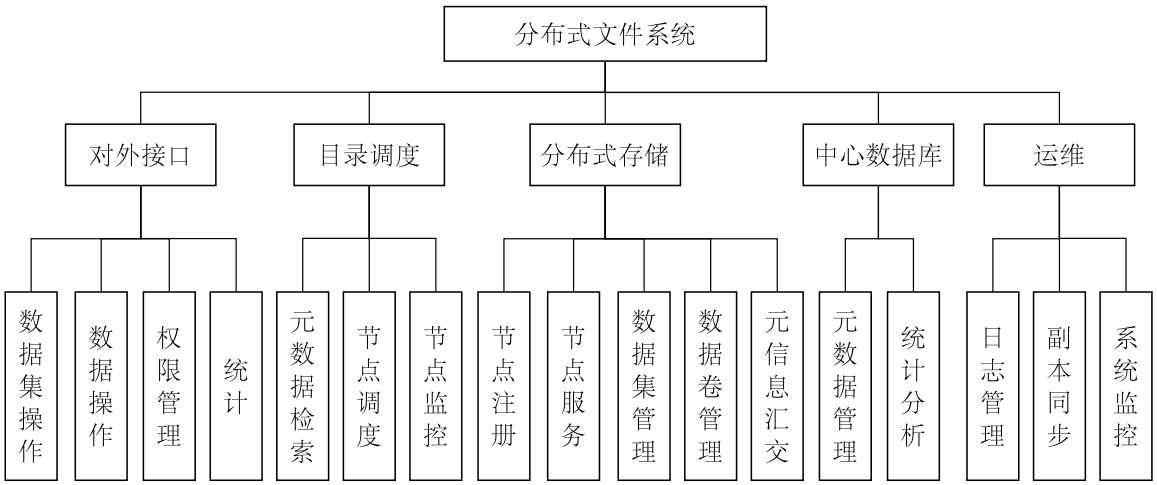


图 16　系统功能模块结构

- 注册类：实现数据集的申请、注册；
- 数据操作类：数据的上传、下载、打包；
- 权限类：数据集权限的管理等；
- 统计类：数据、数据集的统计等。

### 目录调度

目录调度服务是分布式系统的一个核心调度服务，负责将接口的请求进行处理，比如确定节点的位置，文件的信息等。根据请求并且通过接口分发到其他模块进行处理，处理结果再转发给用户。因此这是一个元信息检索定位、存储调度的模块。由于信息都存储在中心数据库以及注册服务数据库中，因此目录调度也可以做成分布式多节点的服务，可以根据需要进行扩充。

### 分布式存储

分布式存储是系统的核心底层服务，由若干存储节点组成。单个节点也可以提供服务。每个节点由多个数据集组成，每个数据集由若干数据卷组成，而每个数据卷则可以存储数千乃至数万的数据文件。

图17是一个分布式存储节点的架构图。

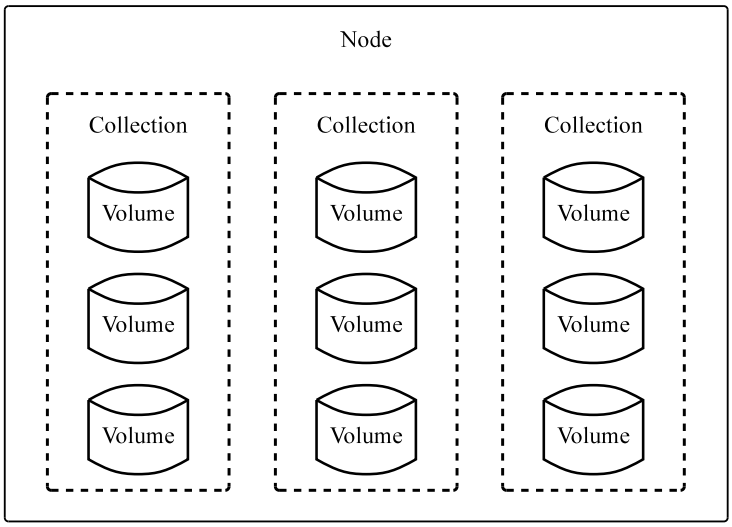


图 17　分布式存储节点

### 中心数据库

中心数据库是一个关系型数据库集群，存储了系统的节点信息、数据集信息、数据卷信息、数据文件信息以及相关的日志信息等。中心数据库通过一个接口实现对数据库的操作，而不是直接操作数据库。

### 运维

运维模块是一个独立的服务，主要的功能是定期的日志管理汇交，数据集、数据卷等的副本同步等。

## 4.3　高可用

作为一个分布式系统，高可用是非常关键的要求，从整个系统架构看，高可用在各个节点都有体现：

- **对外接口**可以使用 nginx 进行负载均衡；
- **目录调度**由于服务是无状态的，因此也是可以做成分布式服务的，可以使用多台机器进行负载均衡；
- **分布式存储**分布式存储节点是互备的，默认以数据集为单位进行三副本互备。
- **中心数据库**可以使用关系型数据库集群；
- **运维**后台任务，可以使用并发。

## 4.4　数据组织

在系统的设计与实现中，天文数据的组织也非常重要，因此，非常有必要重点设计下数据组织的原则和标准：

- 系统管理的数据按数据集为单元存储，每个数据集为一个逻辑上一致的数据组织；
- 划分数据集的原则是按照望远镜的观测计划、日常观测、发布版本或者自定义进行划分，比如一个观测计划可以划分为一个数据集；
- 数据副本以数据卷为基本单元进行副本的同步和复制，一般情况下，当一个数据卷写满之后，就可以进入只读模式，然后同步和复制；
- 每个数据块都需要存储详细元数据。

依据这些原则，可以基本满足日常数据的组织。一般情况下，在望远镜观测中心的数据节点，存储的是本地望远镜的一到多个数据集。在主数据中心存储所有的数据集。

## 4.5　软硬件环境

### 4.5.1　软件环境

**开发语言** Go

“Go 是一种开放源代码的程序设计语言，它意在使得人们能够方便的构建简单、可靠、高效的软件”[29][1]

Go 语言的发明人是 Robert Griesemer、Rob Pike 和 Ken Thompson，其中 Ken Thompson 还是 C 语言的发明人。Go 语言形成构想于 2007 年 9 月，并于 2009 年 11 月发布，

[1]https://golang.org/

目前最新版本为 1.8。Go 特别适合构建基础设施类软件，当前在云计算、大数据基础软件领域的几个重量级软件，诸如 Docker[2]、Kubernetes[3]等都基于 Go 语言构建。2016 年更是第二度成为年度程序开发语言[30]。

Go 语言在近几年的发展中也积累了大量的基础软件库，并且在大量的软件中得到了应用。

系统采用 Go 语言作为主要开发语言基于其以下优点：

- 系统级编程语言；
- 快速编译与执行，目前版本是 1.8，其编译速度更快；
- 基于 CSP 模型的高性能并发程序设计模型；
- 简约的部署模式，可以针对不同的操作系统生成目标可执行程序，而且可执行程序不包含依赖库，直接的拷贝就可实现部署；
- 构建分布式系统所需的一系列基础软件库都得到了生产系统的检验，具有很好的可靠性。

下面是一些构建分布式系统所需的基础软件库：

KV **数据库** Bolt[4]

Bolt 是一个纯 Go 语言编写的键/值存储数据库，灵感来自于 Howard Chu 的 LMDB 项目，目标是为不需要完整的关系型数据库服务的项目提供一个简单、快速、可靠的数据库。Bolt 比较偏底层功能，因此可以根据需要进行功能性的开发。目前已有多个项目的底层使用了 Bolt，比如 Etcd[5]、InfuxDB[6]等。

Bolt 在系统中主要存储底层的元信息，比如文件索引、数据卷索引、数据集索引等。

**关系型数据库** PostgreSQL[7]

PostgreSQL 是一个强大的，开源的对象-关系数据库系统。经过了 15 年的积极开发，它已经拥有了成熟的架构，使其在可靠性，数据完整性和正确性方面赢得了良好的声誉。

**分布式键值存储** Etcd

Etcd 是一个分布式的可靠键值存储系统，使用了 Raft 一致性协议开发。是一个高可用强一致性的服务发现存储服务。主要用于存储分布式系统的关键数据，与 Zookeeper 类似。在系统中的作用是存储节点的信息，一个典型的应用场景是：当一个节点服务启

[2]https://www.docker.com/
[3]https://kubernetes.io/
[4]https://github.com/boltdb/bolt
[5]https://github.com/coreos/etcd
[6]https://www.influxdata.com/
[7]https://www.postgresql.org

动的时候，会向 Etcd 发送注册信息，告诉分布式系统其状态，调度服务会根据在 Etcd 中查询到的节点状态分配任务给节点。这是分布式系统的一个关键服务。

**分布式消息总线** NSQ[8]

NSQ 是实时的分布式消息处理平台，其设计的目的是用来大规模处理数以十亿计级别的消息。NSQ 具有分布式和去中心化拓扑结构，该结构具有无单点故障、故障容错、高可用性以及能够保证消息的可靠传递的特征，是一个成熟的，并且已经在大规模生产环境下应用的产品。

### 4.5.2　硬件环境

硬件环境主要采用 Linux 服务器，磁盘存储环境试验 XFS 文件系统。

## 4.6　小结

本章详细介绍了系统的总体架构，包括多节点系统、主系统、小系统（桌面系统），并且描述了系统的主要模块结构, 最后介绍了系统的软硬件结构，以及需要的一些第三方应用、开发语言等。从总体的角度全面的设计了系统。

[8]http://nsq.io/

# 第五章　系统的详细设计与实现

在前一个章节中，本文对系统进行了总体的设计，并说明了系统的关键技术和数据交互流程。在系统中，核心的关键技术是底层的存储引擎和层次管理模式，底层的存储引擎规划了数据如何存储，层次化规划了数据如何流动与交互。因此，本章将重点说明基于数据流模式的分层结构和底层存储引擎。

## 5.1　基于数据流的关键技术

根据从图6描述的天文数据的全生命周期的分析和提炼，数据流可以分为两类：上传和下载，这是两个相反的数据流向，面向的也是不同的应用场景。

数据上传流：经过望远镜、望远镜数据中心、系统数据中心，最后在备份中心长期保存，系统数据中心也是在线的服务中心。

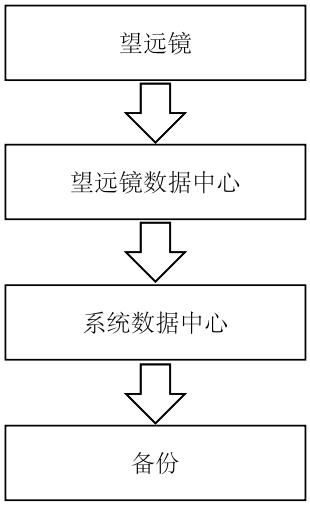


图 18　数据上传数据流

数据下载流：主要是数据中心，面向的是用户，由用户发起请求，下载数据至其要求的地方。

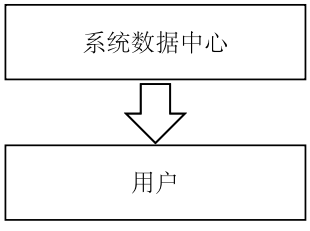


图 19　数据下载数据流

因此，针对业务的需求，数据读写流程是系统的一个核心关键流程，也是一个关键技术需要设计和实现，下面是对此的详细设计。

### 5.1.1　数据存储流程

数据存储流程是数据上传至分布式存储的过程，如图20，共分为下面几个步骤：

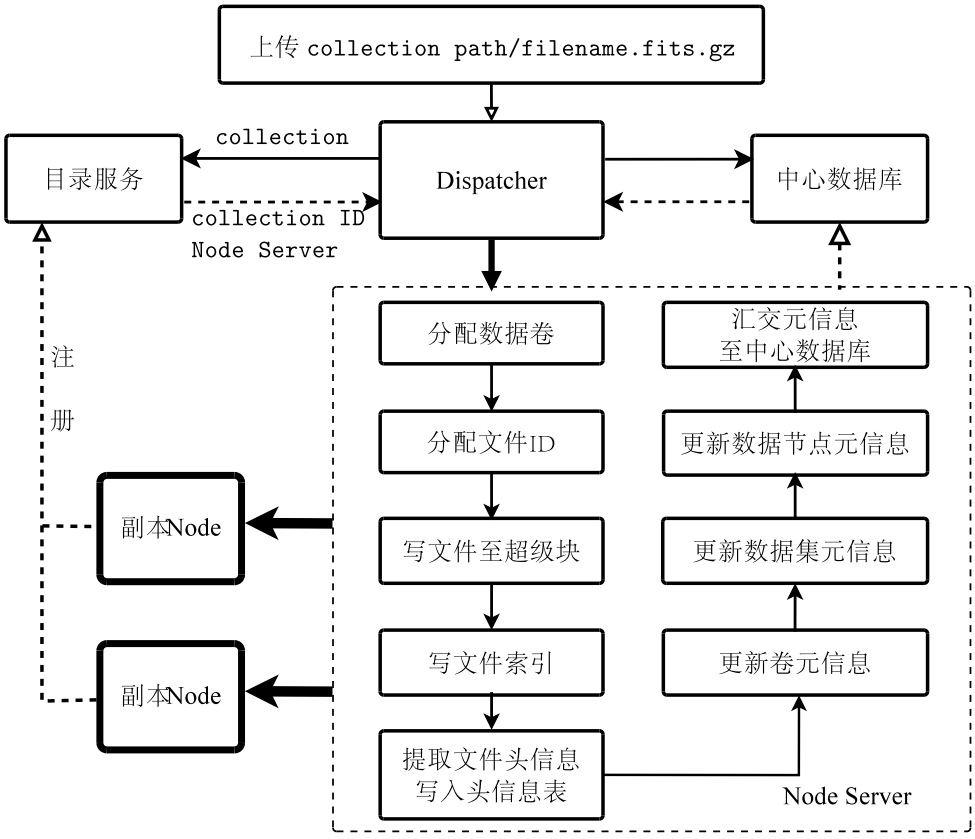


图 20　数据存储流程

1. 客户端提交文件上传请求，参数为数据集名称（collection）和包含文件名称的路径；
2. 调度程序（Dispatcher）查询目录服务，获取数据集的 ID 和节点的服务信息；
3. 调度程序将请求提交给数据节点，数据节点开始：
   （1）　分配将要存储数据卷；
   （2）　分配 ID；
   （3）　写文件到超级块（SuperBlock）中；
   （4）　写文件索引到索引文件中；
   （5）（可选）提取文件头信息到头信息表中；
   （6）　更新卷元信息，添加文件信息到卷信息中；
   （7）　更新数据集元信息；

（8） 更新数据节点元信息；

（9）（定期）汇交数据集元信息到中心数据库。

4. 一个卷写结束后，自动副本到其他指定的副本节点。

### 5.1.2 数据读取流程

数据读取流程，如图21，共分为下面几个步骤：

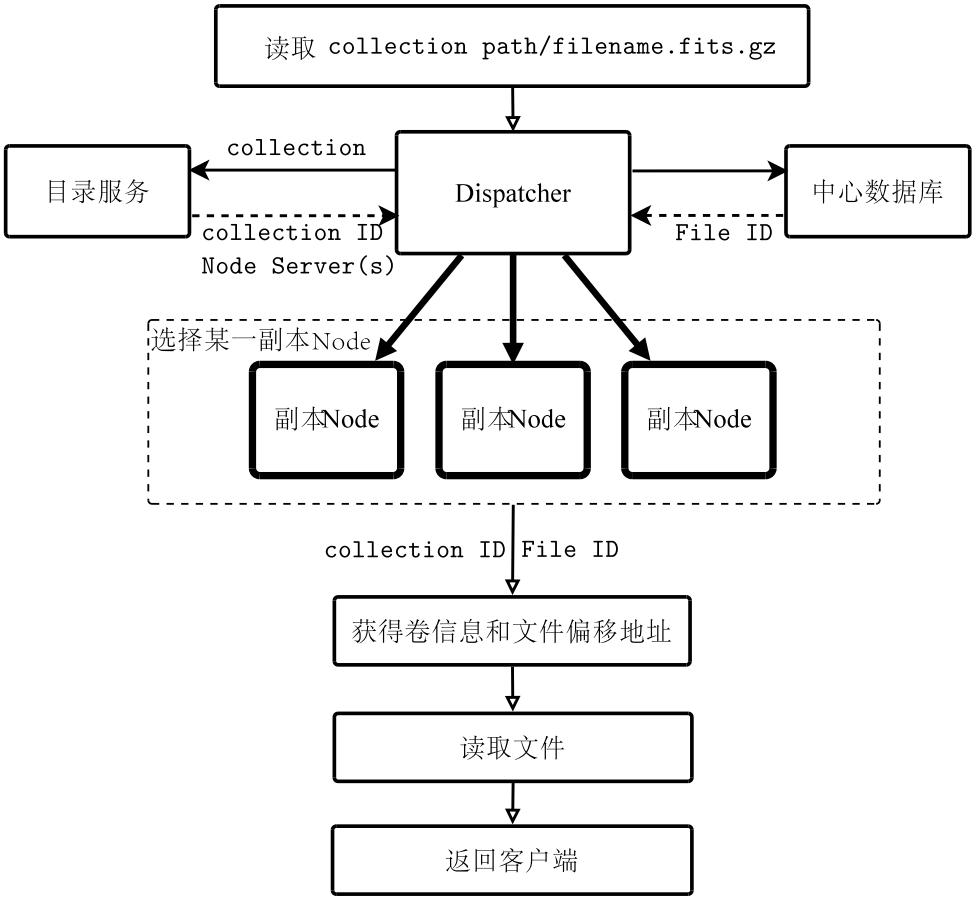


图 21 数据读取流程

1. 客户端提交文件下载请求，参数是参数为数据集名称（collection）和包含文件名称的路径；
2. 调度程序（Dispatcher）查询目录服务，获取数据集的 ID 和节点的服务信息；
3. 调度程序（Dispatcher）查询中心数据库，获取文件的 ID；
4. 根据数据集 ID、文件 ID，选择一个数据节点提交数据访问请求；
5. 数据节点读取文件；
6. 数据节点将数据返回给调度程序；
7. 调度程序返回给客户端。

### 5.1.3　数据副本与优化

数据默认为三副本，当前未考虑使用更复杂的算法进行存储空间的优化。三副本分别位于不同的节点上，读取数据时，系统的调度服务会根据轮询原则来分担数据读取的压力。数据写入时，会先写入一个主节点，写入结束后，由运维模块同步数据至其他节点。

### 5.1.4　分布式与单机模式

系统在设计时就考虑了主系统和小系统（单机模式），这样做的好处是在不同的规模层次使用时，可以选择不同的模式进行部署，并且小系统可理解为分支节点，可部署在望远镜数据中心、个人服务器、桌面电脑上等。并且同时会与中心节点有数据和信息的交互。

## 5.2　底层存储引擎关键技术

如图22, 底层的分布式存储的结构为一个服务节点（Node）包含多个数据集（Collection），每个数据集由多个数据卷（Volume）组成，每个数据卷由一个超级数据块（SuperBlock）和相关索引文件组成。每个数据超级块由多个基本数据块（Block）组成，数据块是基本的数据单元，存储的是真实的完整数据体，也可以理解为对外表示的一个文件。

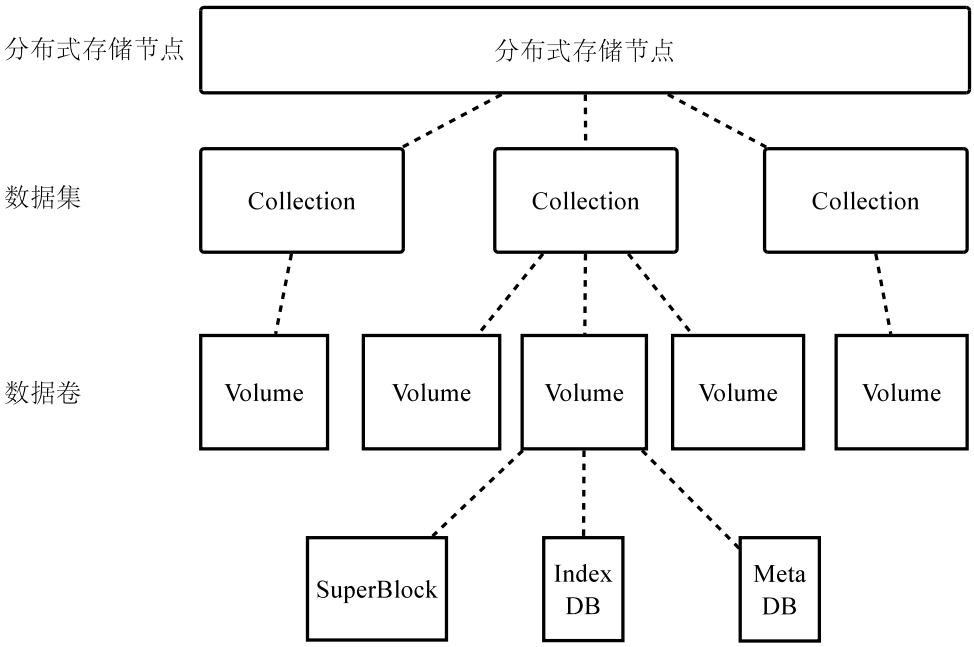


图 22　分布式存储节点的层次结构

以下从基本单元数据块（Block）开始设计。

### 5.2.1 数据块 Block

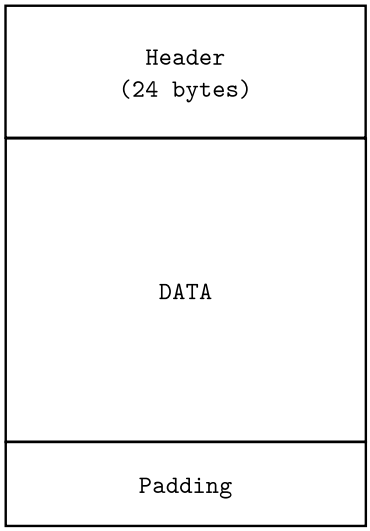


图 23 数据块基本结构

数据块 Block 是系统最基本的原子单元，如图23，数据块主要由三部分组成，其中前 24 个字节是包含了块的基本信息：比如文件 ID、文件删除标记、文件压缩类型（默认为 gzip 格式)、文件的类型、数据块大小、校验码、数据体大小。接下来一部分是数据体完整内容。最后一部分是一段不定长的字节，字节数不大于 8，以此来保证完整的数据块是 8 字节的整数倍，用来提高磁盘的读写性能。

详细的字段格式说明和定义见表2。

表 2 数据块 Block 格式定义

| | 字段 | 位 | 字节数 | 说明 |
|---|---|---|---|---|
| Header | ID | 0 – 7 | 8 | 数据块 64 位唯一编号 |
| | flag | 8 | 1 | 删除标注 |
| | compress | 9 | 1 | 数据压缩类型 |
| | type | 10 – 11 | 2 | 数据块类型（文件类型） |
| | bsize | 12 – 15 | 4 | 数据块大小 |
| | size | 16 – 19 | 4 | 数据体大小 |
| | checksum | 20 – 23 | 4 | 校验码 |
| DATA | data | | 数据体真实大小 | 数据体 |
| Padding | padding | | 0-7 | 8 位对齐 |

1. 字段 ID，也就是文件的 ID，长度为 64 位，理论上可以有 $2^{64}$ 个数据块编号，为了构建一个唯一的 ID 系统，我们建立了如表3的 ID 规范，64 位长整数的低位序号为 0，

最高位序号 63。ID 由三部分组成，分别是数据集编号、卷编号和卷内的流水号。

表 3　数据块 ID 规范

| 位 | 位数 | 值范围 | 说明 |
|---|---|---|---|
| 32 – 63 | 32 | | Collection 编号，参看表5 定义 |
| 20 – 31 | 12 | 0 – 4095 | 卷 ID 编号 |
| 0 – 19 | 20 | 0 – 1048575 | 卷内部顺序号 |

2. 字段 flag 是数据删除标记，如果该字段为 0，表示该数据块是无效的，可理解为已被删除。运维模块会定期的扫描卷，如果一个卷内的被删除数据块较低的时候，会将该数据卷重建，彻底删除无效的数据块。

3. 字段 compress 是数据压缩类型，默认是 gzip 压缩类型，但可选压缩类型还有 LZ4、Snappy 等[31]。一般来说 gzip 的压缩比是最高的，但在输入输出场合，其压缩所需的 CPU 和耗时则是最大的。业界实际应用方面，比如 Google 开源的 Snappy[1]和 Facebook 开源的 Zstandard[2]等都是在压缩比和 CPU 消耗等方面尽量都得到平衡，其主打都是实时压缩。这种情况下虽然压缩比不如 gzip，但性能却有很大提升。

但是根据天文数据的使用场景，实时性要求不是那么高，大量数据都是一次性写入，因此完全可以压缩好再写入数据块中，而一般的天文软件也支持直接读入 gzip 压缩后的数据。因此系统选择 gzip 作为默认的数据压缩选型。至于其他压缩类型的使用场景，则可根据未来业务场景的发展再做调整，如果在实时性计算方面有比较高的要求时，也可以改用 Snappy 等方案进行数据等压缩。系统已经预留了接口，只需要修改配置即可。

4. 字段 type 是数据块存储数据类型（文件类型），是数据文件的真实类型，类型有 FITS、CSV、PNG 等。

5. 字段 bsize 是数据块的大小字节数。

6. 字段 size 是数据体的大小字节数，也就是 data 字段的长度。

7. 字段 checksum 是 data 字段的奇偶校验码。

8. 字段 data 是真正的数据体内容，长度为 size 字段值。

9. 字段 padding 是字节对齐，长度为 0-7 个空字节，以保证整个数据块的总长度是 8 的倍数。

[1]https://github.com/google/snappy

[2]https://github.com/facebook/zstd

### 5.2.2 数据卷 Volume

一个数据卷由一个超级数据块、一个卷索引和一个数据卷元数据库（可选）组成。每个数据卷拥有一个唯一 ID，该 ID 由节点按照数据集内部的顺序号进行编码，数据卷的 ID 在数据集中是唯一的，ID 的范围是 `0001 – 4095`。对应于数据卷的文件命名规则是：

- `0001.vol` 数据超级块
- `0001.idx` 卷索引
- `0001.mdb` 卷元数据库（可选）

**超级数据块** SuperBlock，超级数据块是文件的主存储，由一个简单的头信息和若干个数据块组成，基本结构如图24所示。

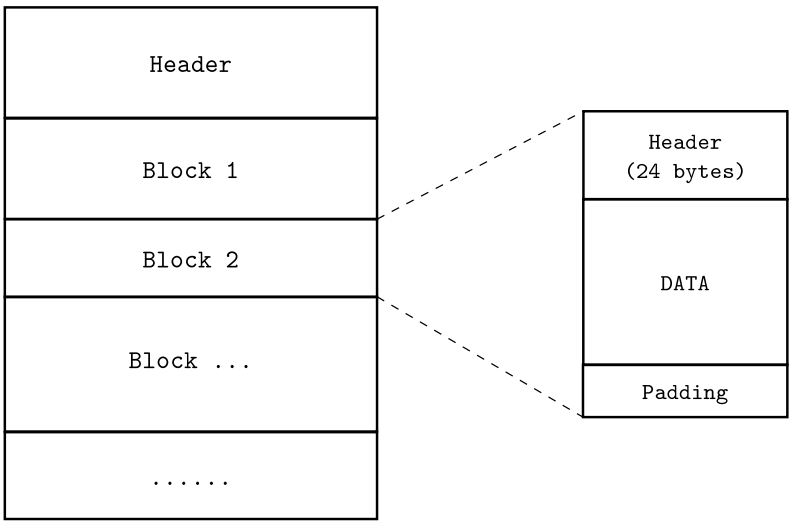


图 24　超级块 SuperBlock 基本结构

超级块中数据块均以追加的模式进行添加，理论上一个卷可以有 1048576（$2^{20}$）个数据块。根据系统的默认配置，一个数据卷超级块的尺寸最大为 100GB，这个参数是可以配置的。

**卷索引**，卷索引主要索引了在超级块中每个数据块的 ID、大小和在超级块中的偏移量，有了这几个参数，很容易对数据块进行快速寻址，卷索引存储在 KV 数据库中。

表 4　数据卷索引结构

| 字段 | 字节数 | 说明 |
|---|---|---|
| `ID` | 8 | 数据块 ID |
| `Offset` | 4 | 数据块的偏移量 |
| `Size` | 4 | 数据块大小 |

**数据卷元数据库**，元数据库主要存储的是文件的一些元信息，比如文件的文件名，文件时间等。另外对于诸如 FITS 等包含元数据的文件格式，将提取这些元数据并且存储至元数据库中。参见图20，元数据的汇交也是从这个数据库中提取信息先汇交到数据集元信息中，再汇交到节点元数据库中，最后汇交到中心数据库中。

数据卷在初始状态都是可读写的，在到达一定阈值后，将进入只读（readonly）模式，不再接受上传写入，并且启动副本传输，同步数据卷到指定的副本节点中。

### 5.2.3　数据集 Collection

数据集（Collection）的定义一般是一个观测计划、或者一个望远镜的观测数据、或者一个逻辑上的一组数据。对于 FITS 文件来说，一般具有相同的 FITS 头信息结构，或者可以理解为，对于一个数据集来说，其元数据的格式是一致的，当汇交到中心数据库中时，表现为一个数据表。

数据集也拥有一个数据集的元数据库，存储来自各个数据卷汇交的信息，比如块信息、索引信息、头信息等。这个数据库将定期向数据节点的元数据进行汇交。

每个数据集拥有一个唯一的 ID 标识，由 4 个字段组成，其定义参见表5。

表 5　数据集 ID

| 字段 | 位 | 位数 | 值范围 | 说明 |
|---|---|---|---|---|
| DataType | 0 – 3 | 4 | 0 – 15 | 数据类型 |
| SN | 4 – 15 | 12 | 0 – 4095 | 顺序号 |
| Telescope ID | 16 – 30 | 15 | 0 – 32767 | 望远镜编号 |
| Country | 31 | 1 | 0、1 | 国家，1 为国内望远镜，0 为国外望远镜 |

一个数据集在文件系统中表现为一个目录，如表5，目录文件夹的名称为数据集 ID 的 16 进制格式。比如数据集 ID 为 12345 的 16 进制为 `0001E240`，因此这个数据集的目录名称即为 `0001E240`。

### 5.2.4　数据节点 Node

数据节点为一个完整的服务节点。当一个 Node Server 服务启动的时候，将启动对该节点数据管理等服务。

数据节点的目录结构：默认系统数据目录的根节点为`/xingfs`，这个目录也是可以配置的。图25就是描述了数据节点的树状目录层次结构。

**数据集目录：**

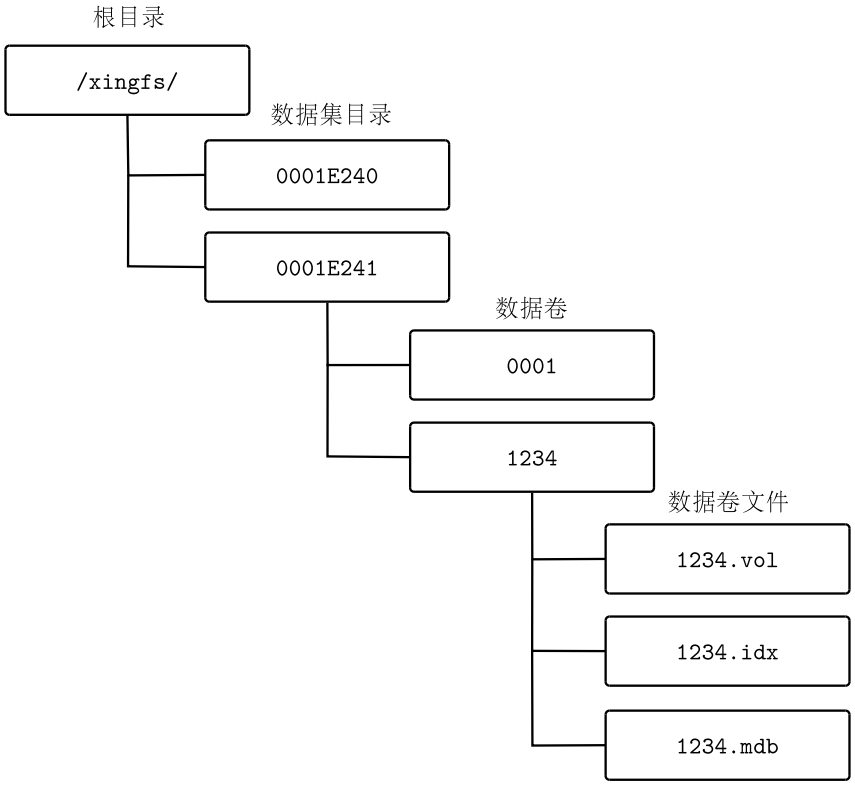


图 25　数据节点的文件存储结构

- `/xingfs/0001E241/` 目录;
- `/xingfs/0001E241/collection.db` 数据集元信息数据库，存储数据集结构和信息。

**数据卷：**

卷 ID 是 0 – 4095 的一个数字，数据卷数据集信息数据库进行顺序号的管理和分配。卷的名称为卷 ID 的十六进制字符表述。例如:

- `/xingfs/0001E241/1234.vol` 卷的超级块;
- `/xingfs/0001E241/1234.idx` 卷索引;
- `/xingfs/0001E241/1234.mdb` 卷元数据库。

## 5.3　数据库体系设计

系统数据库体系是一个层级的体系，如图26所示，共分为四层，底层是数据卷的数据库，往上是数据集的数据库，再往上是节点数据库，最高层是中心数据库，其中底下的两层可以使用 KV 数据库进行存储，上面两层可以使用关系型数据库进行数据存储。从整个汇交体系上可以看到，每一层都是上一层的一个子集，通过数据的层层汇交，可以实现模块的解耦和数据的高可用。

以上是数据的增量处理的模式，对于数据的删除来说，其流程是将数据块的 flag 置为 0，然后逐级在各层次的数据库中删除记录。对于数据的更新来说，则是先执行删除，

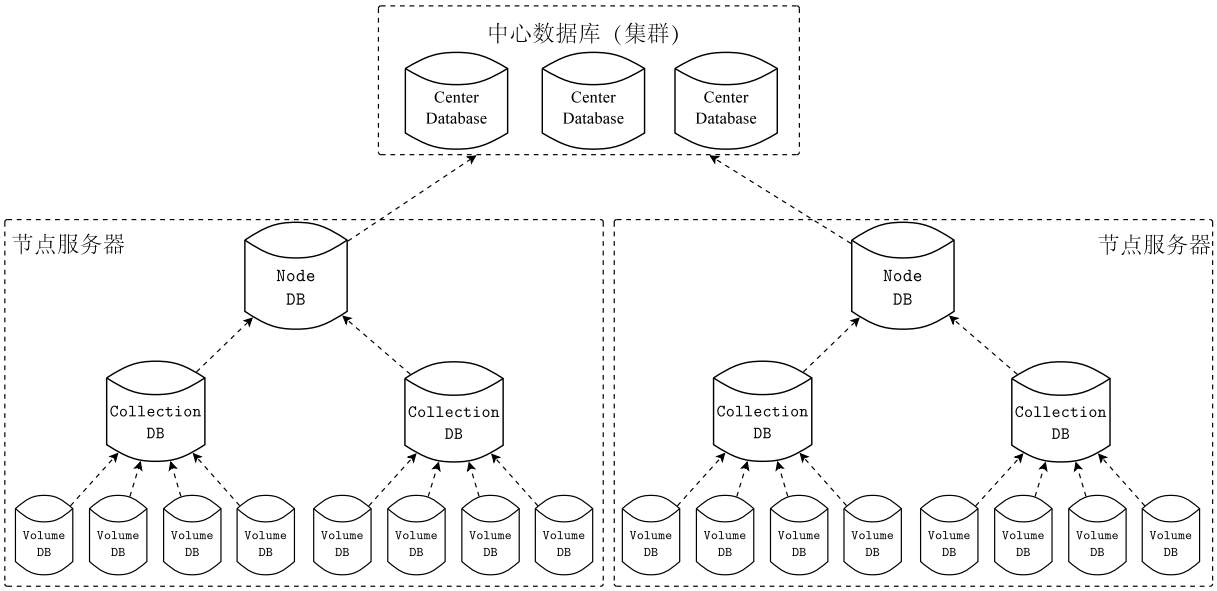


图 26　数据库数据汇交流程图

再追加新的纪录。

运维系统会定期进行评估，当某个卷的被删除文件量占卷大小的比例小于某个阈值时，进行卷更新操作，卷更新操作是将原卷的有效数据复制到新卷中后，然后删除原卷。

### 5.3.1　中心数据库

中心数据库可以是一个高性能关系型数据库，根据数据量的规模，也可以建设为高可用的集群数据库。目前采用的是 PostgreSQL 数据库系统的技术方案。

### 5.3.2　节点数据库

节点数据库存储的是节点的信息，并且存储了节点下所有数据集、数据卷、数据块的信息，这些信息以准实时模式向中心数据库进行同步和汇交。节点数据库在部署模式上使用的也是一个关系型数据库。

数据集数据库存储的信息有：

- 节点内所有数据集内所有卷内块索引表，如表4；
- 节点内所有数据集内所有卷内块详细信息表；
- 节点统计表；
- 节点内所有数据集内所有卷内块头信息表（如果是 FITS 文件，则为头信息）。

### 5.3.3 数据集数据库

数据集数据库存储的是数据集等元信息以及数据集中数据卷、数据块的信息。这些信息以准实时模式向节点数据库进行同步和汇交。由于数据集数据库并不直接与服务系统交互，因此可以使用 KV 数据库进行存储和持久化。

数据集数据库存储的信息有:

- 数据集内所有卷内块索引表，如表4;
- 数据集内所有卷内块详细信息表;
- 数据集统计表;
- 数据集内所有卷内块头信息表（如果是 FITS 文件，则为头信息)。

### 5.3.4 数据卷元数据库

数据卷元数据库记录的是数据卷的元信息以及数据块的详细信息，这些信息以准实时模式向数据集数据库进行同步和汇交。在这里也是使用 KV 数据库进行存储和持久化。

数据卷数据库存储的信息有:

- 卷内块索引表，如表4;
- 卷内块详细信息表;
- 卷统计表;
- 卷内块头信息表（如果是 FITS 文件，则为头信息)。

### 5.3.5 数据库体系设计小结

整个数据库体系结构的设计思路是“**独立成库，逐层汇交**”，也就是在每一层的数据库设计中，数据库都记录了本层次完备的信息，可以独立成库，这样也是一个模块化的设计。比如当一个数据卷写好之后，将进入只读状态，只需要将该卷的三个文件（超级块、索引、元数据库）同步复制到其他节点，就可以直接实现副本的同步。

逐层汇交可以实现对下一层次的完整性掌控，只需访问该层的数据库就可以知晓本层以下所有的元数据信息。经过这样的层级设计，可以非常好进行数据扩展、数据副本、数据同步等，实现高效的数据操控。

## 5.4 服务与接口设计

系统需要额外部署的软件有数据库系统 PostgreSQL、分布式键值存储系统 Etcd、以及分布式消息服务系统 NSQ。如图27，分布式系统服务端程序由以下 4 个部分组成:

- **Dispatcher** 系统的入口服务，也可以在多节点部署，做成分布式负载均衡。
- **NodeApp** 节点服务程序，根据 Go 语言的特点，可以在一个程序中封装多个同时运行的模块，而且模块之间可以进行通讯。
  - *NodeServer* 对外服务程序，以 RESTful 模式与 Dispatcher 进行通讯；
  - *Store* 管理数据集、数据卷存储的模块；
  - *Ops* 运维模块，负责数据库、元数据库的汇交，以及与消息总线的通讯。
- **DatabaseOps** 数据库运维程序，负责数据库的同步、备份。
- **NodeOps** 节点运维程序，负责进行副本同步、部署、上线等操作。

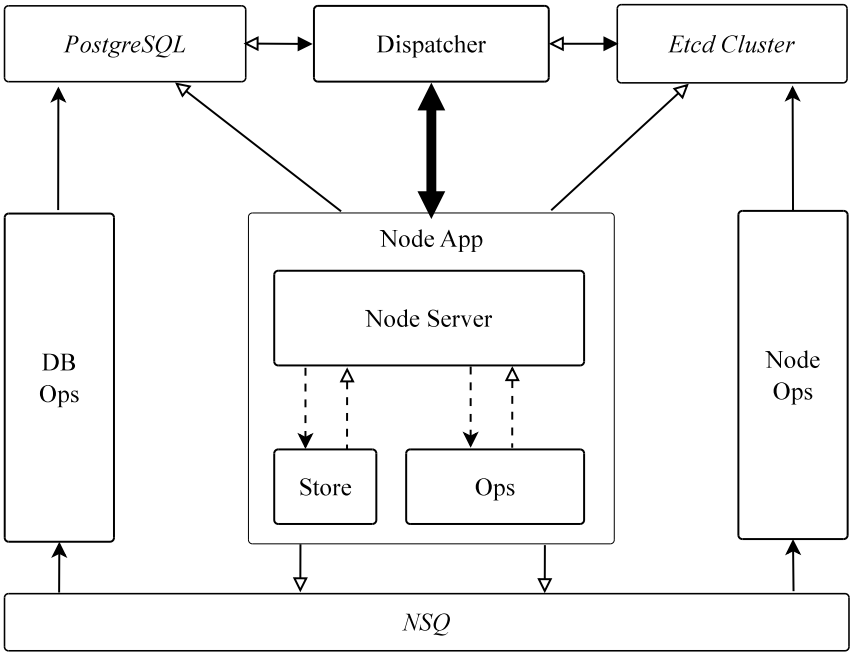


图 27　系统软件模块图

### 5.4.1　Dispatcher

系统对外的接口主要基于 RESTful 模式提供服务，服务的接口类型主要分三个部分：

1. 管理类，处理数据集创建等申请；
2. 数据处理类，这个是最主要的服务，主要处理数据的上传和下载，偶尔也会处理数据的删除操作；
3. 统计类，主要负责生产相关的数据报表。

服务接口通过内部的 HTTPS 通讯与数据节点交互，从各个节点读取数据，提交数据等。

### 5.4.2 NodeApp

节点程序在启动时，会读取配置信息，并且将数据集、数据卷的元信息载入到内存数据库中，以加速数据访问。数据块的索引也会被加载到内存中。加载结束后，会向注册服务器（Etcd）提交注册，表明本节点已经启动，可以对外提供服务。

数据卷的创建也是由节点程序负责，包括分配好数据卷 ID，生成空超级块文件、空索引文件和空元数据文件，并等待文件的写入。当卷的大小达到预设的卷阈值（比如一个卷 100GB）时，结束该卷的写入，建立新的卷写入。

当一个卷完成写入后，该卷将进入只读状态，并通知内部的运维模块（Ops）进行后续的操作：卷的索引和元信息逐级汇交至上级的数据集和节点，另外也通过外部的消息总线通知 NodeOps 进行数据卷的副本操作。

如果是删除或者更新操作，则首先需要将卷设为修改模式。

### 5.4.3 NodeOps

NodeOps 接收来自消息总线的信息，主要是进行卷副本的同步工作。

### 5.4.4 DatabaseOps

DatabaseOps 接收来自消息总线的信息，主要的操作是数据库运维程序，负责数据库的同步、备份。同时根据规划，定期进行数据库的快照等。

### 5.4.5 注册服务接口

注册服务是分布式的一个协调中心，当节点上线或者下线（包括异常下线）时，主程序会通过注册服务知晓哪些节点可用，哪些节点不可用等。

### 5.4.6 消息总线

消息总线提供系统模块间的通信和消息分发功能，比如数据读写结束后的消息通知、副本复制消息等。

## 5.5 系统安全

出于安全、性能以及未来趋势的综合考虑，系统主要使用了 HTTPS 与 HTTP/2 等技术。

首先 HTTPS 系统可以保证数据传输中的安全问题。HTTPS 所需要的证书等都是自己生成和分发，而且进行了定制，分发出去的证书是唯一的，以此保证届时分布的全国

各地的站点节点与主节点的通讯不仅免密、安全，而且可以完全的跟踪。

HTTP/2 改进了之前使用了十多年的 HTTP/1.1 协议的不足，可以提高整个网络的效率，并且这也是未来的趋势，当前最新主流浏览器大多已经堆砌提供支持。另外一点就是 HTTP/2 必须使用 HTTPS。

## 5.6　文件级 API 设计

数据文件的上传下载以 RESTful 模式进行，以下分别是文件上传、下载、删除、查询的基本操作接口：

### 5.6.1　文件上传

表 6　API：文件上传

| POST /xs/:collectionName/:filename | |
|---|---|
| **方法** | POST |
| **路径** | /xs/:collectionName/:filename |
| **参数** | 无 |
| **返回** | 上传时间和文件大小 |
| **相应内容** | 200 |
| **示例** | POST /xs/lamost-raw/20170328/HD081449N430205M02/83290842.fit.gz |

文件上传，提交路径就是数据集的名称和文件的路径。如果的话，返回上传时间和文件大小。

数据集的名称统一由中心节点分配，一般先由子节点（数据节点）申请，数据集名称拥有全局唯一的名称和 ID。

文件的路径为全路径，可以分多级子目录。

### 5.6.2　文件下载

文件下载，根据路径进行文件的下载。如果文件不存在，返回错误信息。

### 5.6.3　文件删除

文件删除，根据路径删除文件，具体在底层是，将 Block 的 flag 字段设置为 0，数据库中相应记录删除。如果文件不存储，返回错误信息。

表 7　API：文件下载

| GET /:collectionName/:filename | |
|---|---|
| **方法** | GET |
| **路径** | /:collectionName/:filename |
| **参数** | 无 |
| **返回** | 文件体或错误信息 |
| **相应内容** | 200 |
| **示例** | GET /lamost-raw/20170328/HD081449N430205M02/83290842.fit.gz |

表 8　API：文件删除

| DELETE /xs/:collectionName/:filename | |
|---|---|
| **方法** | DELETE |
| **路径** | /xs/:collectionName/:filename |
| **参数** | 无 |
| **返回** | 删除成功或者失败 |
| **相应内容** | 200 |
| **示例** | DELETE /xs/lamost-raw/20170328/HD081449N430205M02/83290842.fit.gz |

表 9　API：文件是否存在

| HEAD /:collectionName/:filename | |
|---|---|
| **方法** | HEAD |
| **路径** | /:collectionName/:filename |
| **参数** | 无 |
| **返回** | True 或者 False |
| **相应内容** | 200 |
| **示例** | HEAD /lamost-raw/20170328/HD081449N430205M02/83290842.fit.gz |

### 5.6.4　文件是否存在

文件是否存在，根据路径检查文件是否存在，返回 True 或 False。

## 5.7　客户端

客户端主要实现两个功能，数据上传和数据下载，数据上传，可以实现文件的上传、文件夹的上传等。下载亦然。

### 5.7.1　命令行客户端

命令行客户端是一个预先编译好的客户端程序，需要先配置好参数文件，比如证书的位置、客户端的编号、默认上传数据集的编号、服务端的地址等。

系统提供了一个本地的客户端程序：`xingfs_ctl`，可以实现上述的功能。

### 5.7.2　编程语言接口

由于服务端接口是基于 HTTP 协议的，因此，用户可以根据协议使用自己熟悉的程序语言编写客户端程序或脚本。

## 5.8　小结

本章是系统的一个核心的底层详细设计的一章，详细介绍了系统底层的详细设计，设计了数据存储的层次结构，并且定义了数据存储的数据结构；详细设计了系统数据库的层次架构，这是一个结构清晰的多层结构；还设计与定义了服务与接口的一些模型和规范。

# 第六章　系统测试与部署

## 6.1　系统测试

### 6.1.1　测试目标

系统测试的目标分为两个部分：一是功能性测试，比如底层的存储、分布式系统通讯等，另一个是压力测试，主要测试在不同的压力和网络环境下，系统的性能能否达到要求。

### 6.1.2　系统测试环境

系统测试环境包含了三个数据节点，由三台服务器构成：

- CPU：Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz，64 核
- 内存：132GB
- 操作系统：CentOS 7.2, x86_64
- 存储：15TB

输入数据使用了三个数据集：兴隆 2.16 米望远镜数据、SCUSS 巡天原始数据和 LAMOST 巡天 DR2 产品数据。测试数据规模在 8TB。

一台服务器作为主服务器、另外两台是子节点。

### 6.1.3　系统功能测试

#### 基本功能测试

基本功能测试的目标是测试基本的数据上传下载、建立数据集、备份数据集、统计等功能。测试使用了系统的一个命令行程序 `xingfs_ctl`，图28是程序的基本用法。

1. 建立数据集

建立数据集命令是在当前节点构建一个数据集，命令参数为数据集名称，构建成功后会给新数据集分配一个数据集 ID，并初始化数据集目录结构。测试结果如图29。

2. 上传单个文件

上传单个文件，成功后会返回数据上传的数据卷，分配到文件大小，文件时间等信息。测试结果如图30。

文件上传成功后会返回数据上传的节点 IP，数据块（文件）编号等信息。

3. 单个文件下载

```
[zq@zhuque1 ~]xingfs_ctl

XingFS CTL v1.0

Usage: xingfs_ctl COMMAND [ARGS]...

Commands:
  lc        list collections
  cc        create a collection
  sc        show a collection information
  bc        backup a collection to another node
  uf        upload a file
  df        download a file
  ud        upload all files in a directory
  dd        download all files in a directory
```

图 28　xingfs_ctl 命令行程序

```
[zq@zhuque1 ~]xingfs_ctl cc lamost_raw
----------------------------------------
Node            | 10.2.0.31
Collection      | lamost_raw
Collection ID   | 1001 (000003EB)
Collection Path | /xingfs/000003EB
Volumes         | 0
Size            | 0
Create Time     | 2017-04-23 13:32:06
----------------------------------------
```

图 29　建立数据集

```
[zq@zhuque1 ~]xingfs_ctl uf lamost_raw survey2016/20170422/HD153504N014009B02/ro-05r-
20170423021106-3-83327171.fit.gz
100%[========================================]

------------------------------------------------------------
Node        | 10.2.0.31
Collection  | lamost_raw (000003EB)
Volume      | /xingfs/000003EB/0001.vol
File ID     | 4307853246995
File Name   | ro-05r-20170423021106-3-83327171.fit.gz
File Size   | 17893455
File Time   | 2017-04-23 13:43:00
------------------------------------------------------------
```

图 30　上传单个文件

单个文件下载，会将文件下载到本地，在这里，可以看到文件的时间及大小与上传前是一致的。测试结果如图31。

4. 文件批量上传

文件批量上传可以用来上传一个文件目录，上传成功后，返回文件的基本统计和文件总的大小。测试结果如图32。

```
[zq@zhuque1 ~]xingfs_ctl df lamost_raw survey2016/20170422/HD153504N014009B02/ro-05r-
20170423021106-3-83327171.fit.gz
100%[==========================================]

[zq@zhuque1 ~]ls -l
-rw-rw-r—1 zq archive 17893455 Apr 23 02:43 ro-05r-20170423021106-3-83327171.fit.gz
```

图 31　单个文件下载

```
[zq@zhuque1 ~]xingfs_ctl ud lamost_raw survey2016/20170422/HD153504N014009B02
100%[==========================================]

------------------------------------------------------------
Node        | 10.2.0.31
Collection  | lamost_raw (000003EB)
Volume      | /xingfs/000003EB/0001.vol
File Path   | survey2016/20170422/HD153504N014009B02
Files       | 96
Files Size  | 1886826322
------------------------------------------------------------
```

图 32　文件批量上传

批量上传成功后，还会返回上传了多少文件以及文件的总数据量。

5. 文件批量下载

文件批量下载用于下载一个目录到本地，下载结束后可以看到下载到内容。测试结果如图33。

```
[zq@zhuque1 ~]xingfs_ctl dd lamost_raw survey2016/20170422/HD153504N014009B02
100%[==========================================]

[zq@zhuque1 ~]ls HD153504N014009B02
ro-01b-20170423021106-3-83327171.fit.gz  ro-09b-20170423021106-3-83327171.fit.gz
ro-01b-20170423023928-4-83327199.fit.gz  ro-09b-20170423023928-4-83327199.fit.gz
ro-01b-20170423030750-5-83327227.fit.gz  ro-09b-20170423030750-5-83327227.fit.gz
ro-01r-20170423021106-3-83327171.fit.gz  ro-09r-20170423021106-3-83327171.fit.gz
ro-01r-20170423023928-4-83327199.fit.gz  ro-09r-20170423023928-4-83327199.fit.gz
ro-01r-20170423030750-5-83327227.fit.gz  ro-09r-20170423030750-5-83327227.fit.gz
ro-02b-20170423021106-3-83327171.fit.gz  ro-10b-20170423021106-3-83327171.fit.gz
ro-02b-20170423023928-4-83327199.fit.gz  ro-10b-20170423023928-4-83327199.fit.gz
ro-02b-20170423030750-5-83327227.fit.gz  ro-10b-20170423030750-5-83327227.fit.gz
......
```

图 33　文件批量下载

6. 数据集查询

数据集查询当前节点数据集的基本信息。测试结果如图34。

查询成功后，返回数据集等基本信息，比如有多少数据卷，多少数据量等信息。

7. 数据集备份

```
[zq@zhuque1 ~]xingfs_ctl sc lamost_raw
----------------------------------------
Node            | 10.2.0.31
Collection      | lamost_raw
Collection ID   | 1001 (000003EB)
Collection Path | /xingfs/000003EB
Volumes         | 1
Size            | 1886826322
Update Time     | 2017-04-23 16:12:17
----------------------------------------
```

图 34　数据集查询

数据集备份一般在一个数据卷写完之后，数据卷进入只读状态时，其可以同步到其他节点去，也可以手动同步，如图35的指令。

```
[zq@zhuque1 ~]xingfs_ctl bc lamost_raw 10.2.0.32
100%[==========================================]

----------------------------------------
Node From       | 10.2.0.31
Node To         | 10.2.0.32
Collection      | lamost_raw
Collection ID   | 1001 (000003EB)
Collection Path | /xingfs/000003EB
Volumes         | 1
Size            | 1886826322
Update Time     | 2017-04-23 17:41:07
----------------------------------------
```

图 35　数据集备份

**正确性测试**

测试数据文件入库后，然后下载，对比原始文件和下载文件两组文件是否一致。如图36，在这里测试了 1000 万个随机文件，每个文件大约 10-100KB，随机生成，总共约 437GB，首先将文件生成到 src 目录并且上传，然后再下载到 test 目录，然后对每对文件进行比对，确认是否一致，经过检测，文件是一致的，符合设计时数据完整性的要求。

### 6.1.4　容错性测试

输入参数为异常或者错误时，应该有预期的输出。以下是测试情况。

1. 重复新建一个数据集

新建一个数据集，但数据集名称已存在，返回错误信息，测试结果如图37。

2. 上传本地不存在的数据

```
[zq@zhuque1 test]./checkdiff.sh
Upload:   100%[==================================]
Download: 100%[==================================]

测试对比
-------------------------------------------------------
Source            | Test                | 是否一致
-------------------------------------------------------
src/00000001.fits | test/00000001.fits  | 是
src/00000001.fits | test/00000001.fits  | 是
src/00000001.fits | test/00000001.fits  | 是
......
......
src/09999999.fits | test/09999999.fits  | 是
src/10000000.fits | test/10000000.fits  | 是
-------------------------------------------------------

测试结果：数据一致性 100%
```

图 36　正确性测试

```
[zq@zhuque1 ~]xingfs_ctl cc lamost_raw
----------------------------------------
Status  | fail
Message | 该数据集已注册，数据集编号 1001
----------------------------------------
```

图 37　新建数据集失败，返回错误信息

本地文件不存在，返回错误信息，本地测试结果如图38。

```
[zq@zhuque1 ~]xingfs_ctl df lamost_raw ro-05r-20170423021106-3-83327171.fit.gz
-------------------------------------------------------------------------------
Status  | fail
Message | 文件不存在，文件路径 lamost_raw ro-05r-20170423021106-3-83327171.fit.gz
-------------------------------------------------------------------------------
```

图 38　上传本地不存在的数据

3. 服务器空间不足

数据存储节点空间不足，返回错误信息，本地测试结果如图39。

```
[zq@zhuque1 ~]xingfs_ctl ud lamost_raw survey2016/20170423/HD104231N181311M02
-------------------------------------------------------------------------
Status  | fail
Message | 服务器空间不足：文件夹大小 1878769861，服务器空间剩余 475MB
-------------------------------------------------------------------------
```

图 39　服务器空间不足

4. 下载文件不存在

请求的路径下文件不存在，返回错误信息，本地测试结果如图40。

```
[zq@zhuque1 ~]xingfs_ctl df lamost_raw ro-05r-20170423021106-3-83327171.fit.gz
--------------------------------------------------------------------------------
Status  | fail
Message | 文件不存在，文件路径 lamost_raw ro-05r-20170423021106-3-83327171.fit.gz
--------------------------------------------------------------------------------
```

图 40　下载文件不存在

### 6.1.5　压力测试

测试 100 万个 1K 文件的文件上传、下载的操作，测试响应时间。测试方式是编写了测试脚本进行。100 万个 1K 文件为脚本实时生成的随机文件，内容为随机内容。表10是测试结果：

表 10　压力测试

| 测试项目 | 测试目标 | 耗时（秒） |
|---|---|---|
| 数据上传 | 在本机环境下，测试上传 100 万个 1K 文件 | 152.441 |
| 数据下载 | 在本机环境下，测试下载 100 万个 1K 文件 | 148.326 |

### 6.1.6　测试结论

通过上述对系统的测试，可以看到系统已经可以满足日常的基本需求，达到预计的设计目标，可以为用户提供数据上传、下载等基本服务。

## 6.2　系统实施与部署

系统作为一个分布式的系统，多个模块都可以以分布式模型部署，另外由于系统的灵活性，可以抛开分布式的模式，以单机版模式进行部署。

### 6.2.1　主节点部署

主节点部署的方式如下：

#### 基础环境

- PostgreSQL 数据库
- Etcd 分布式键值数据库

- Nginx 负载均衡

#### 主服务

主服务是系统对外的出入口。如图41，主节点的部署由基础环境加上 Dispatcher、Directory 以及 Matser NodeServer 等服务器完成。

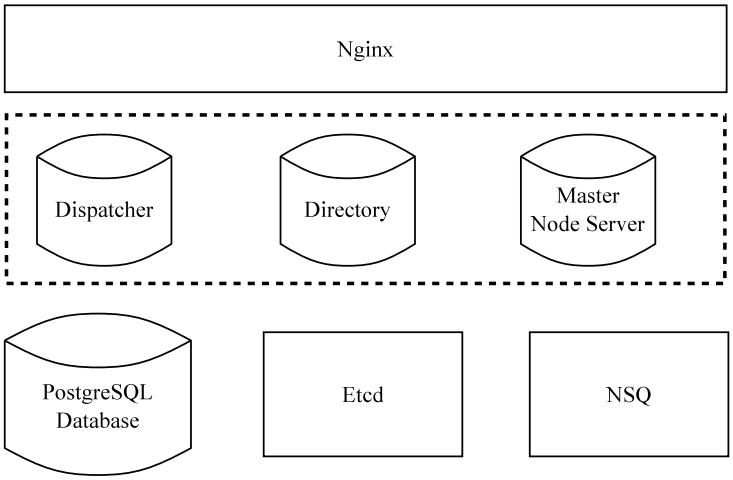


图 41　主节点部署图

#### 子节点

若干台服务器，每个服务器上运行子节点程序 NodeServer。子节点程序的部署极为简单，仅需要将二进制可执行程序部署到启动程序中，同时配置好环境参数，比如节点编号，主节点 IP、端口，数据存储目录等信息，启动服务即可。

由于是个分布式系统，因此需要提前规划好 IP、存储、数据库等基础信息。

#### 运维

运维模块已经包含在主节点程序和子节点程序中，会在程序启动的时候自动启动并开始工作。

### 6.2.2　小系统部署

小系统是一个裁剪的系统，可以选择使用 PostgreSQL 或者 SQLite 作为关系型数据库存储元数据。Go 语言的丰富特性可以仅仅将二进制文件复制到计算机上运行就可以完成部署。

小系统将主系统的各个服务集成到一个程序内，利用 Go 语言的 goroutine 特性组合，形成一个独立的程序。

## 6.3　小结

本章设计了一些测试用例，进行了基本的功能和性能方面的测试，从测试中可以看出，系统可以满足用户对数据管理方面的需求。最后简单的介绍了系统部署的模式。

# 总结与展望

## 总结

本文从国内望远镜尤其是近些年高速发展的大型望远镜的天文数据规模和管理出发，详细介绍了背景和需求，并且列举了一些国内外研究进展以及工业界的一些解决方案，并对需求进行了详细的分析。

第四章开始对系统进行了详细的总体设计，并且在第五章进行详细细致的具体设计，包括数据流程、系统拓扑结构、数据存储结构，ID 规则、数据库层级等。这两章是本文等核心设计。

第六章是系统的测试和部署，系统的测试分为三个层次和部分，有核心的存储，有小系统和大系统的测试，以及分布式测试。系统的部署，系统的部署应用是本文的一大特色，这也是在前面几章点出的一个系统设计时的思想——模块化，模块化不仅体现在系统总体上，也体现在底层的存储上，底层的存储，只需要将一个数据集的目录复制到另外一个节点，甚至是一个桌面电脑上就可以实现一个小系统的访问管理，极大的方便了用户的使用体验。

本文主要的设计思想和设计目标是构建一个针对天文海量数据的分布式数据管理平台，以建立一个完整的底层文件型数据存储实现为基本目标，不追求高的复杂度和花哨多余的功能。

在系统的设计与实现中，本人主要的工作有：提出了基于分布式文件系统的系统设计思路；完成了系统的总体设计和主要的详细设计方案，设计了详细的数据流程；设计了测试的方案；也完成了主要以及核心的编码工作。

## 展望

在现在乃至一个很长的时期内，人工智能将逐步在各个领域全面开花，2016 年可以称作是人工智能元年。在天文领域，数据的高速发展带来的数据暴涨也为数据处理带来了巨大的挑战，这个系统只是一个开始而已，其目标是作为天文海量数据等基础设施，必将在未来面临更多的应用和挑战，也会随着天文海量数据共同发展。

因此功能的拓展也是未来系统演化发展的一个方向，比如更加易用的界面，包括 Web 界面、手机 App 界面等，都已经被考虑在接下来的版本中实现。

## 结束语

中国虚拟天文台在近些年的发展中逐渐从“虚拟”走向“实体”，各种应用蓬勃发展，但摆在我们面前的道路还很漫长，需要做的工作还很多。基于分布式的天文海量数据管理系统只是其中的一块基石，它的真正落地和应用尚需付出更多的努力。但是我相信，通过中国虚拟天文台团队的共同努力，以及与国内外同行、计算机专家的精诚合作，未来一定更美好。

# 参 考 文 献


[1] Wicenec A, Knudstrup J, Johnston S. ESO's Next Generation Archive System[C] // Astronomical Data Analysis Software and Systems XI : Vol 281. 2002 : 95.

[2] Dobrzycki A, Brandt D, Giot D, et al. Observations Metadata Database at the European Southern Observatory[C] // Astronomical Data Analysis Software and Systems XVI : Vol 376. 2007 : 385.

[3] Jurić M, Kantor J, Lim K, et al. The LSST Data Management System[J]. arXiv preprint arXiv:1512.07914, 2015.

[4] Pence W D, Chiappetti L, Page C G, et al. Definition of the flexible image transport system (FITS), version 3.0[J]. Astronomy & Astrophysics, 2010, 524 : A42.

[5] Hanisch R, Quinn P. The International Virtual Observatory[EB/OL]. . http://ivoa.net/about/TheIVOA.pdf.

[6] 李建, 崔辰州, 何勃亮, 等. 天文数据库回顾与展望 [J]. 天文学进展, 2013, 1 : 002.

[7] 崔辰州, 赵永恒. 中国虚拟天文台体系结构 [J]. 天文研究与技术: 国家天文台台刊, 2004, 1(2) : 140 – 151.

[8] 崔辰州. 中国虚拟天文台系统设计 [J]. 博士论文, 中国科学院研究生院, 2003.

[9] 崔辰州, 薛艳杰, 李建, 等. 虚拟天文台——天文学研究的科研信息化环境 [J]. 中国科学院院刊, 2013, 28(004) : 511 – 518.

[10] 张彦霞, 崔辰州, 赵永恒. 21 世纪天文学面临的大数据和研究范式转型 [J/OL]. 大数据, 2016, 2(6) : 2016067. http://www.infocomm-journal.com/bdr/CN/abstract/article_158036.shtml.

[11] Borne K, Accomazzi A, Bloom J, et al. Astroinformatics: A 21st Century Approach to Astronomy[C] // astro2010: The Astronomy and Astrophysics Decadal Survey : Vol 2010. 2009.

[12] Borne K D. Astroinformatics: data-oriented astronomy research and education[J]. Earth Science Informatics, 2010, 3(1-2) : 5 – 17.

[13] 崔辰州, 于策, 肖健, 等. 大数据时代的天文学研究 [J]. 科学通报, 2015(5) : 445 – 449.

[14] Feigelson E D, Babu G J. Big data in astronomy[J]. Significance, 2012, 9(4) : 22 – 25.

[15] 杨传辉. 大规模分布式存储系统 [M]. 北京 : 机械工业出版社, 2013.

[16] Depardon B, Le Mahec G, Séguin C. Analysis of six distributed file systems[J], 2013.

[17] Beaver D, Kumar S, Li H C, et al. Finding a Needle in Haystack: Facebook's Photo Storage[C] // Proceedings of the 9th Symposium on Networked Systems Design and Implementation (OSDI' 2010) : Vol 10. 2010 : 1 – 8.

[18] 储晓颖. 经典论文翻译导读之《Finding a needle in Haystack: Facebook's photo storage》[EB/OL]. (2013/03/07) [2017/02/03]. http://www.importnew.com/3292.html.

[19] 毛剑. bfs: 支撑 Bilibili 的小文件存储系统 [EB/OL]. (2016/04/11) [2017/02/03]. https://mp.weixin.qq.com/s?__biz=MzAwMDU1MTE1OQ==&mid=406016886&idx=1&sn=f5aa286373fb981c9de904568fe7ddb2.

[20] Ghemawat S, Gobioff H, Leung S-T. The Google file system[C] // ACM SIGOPS operating systems review : Vol 37. 2003 : 29 – 43.

[21] Cui C, Yu C, Xiao J, et al. AstroCloud, a Cyber-Infrastructure for Astronomy Research: Overview[C] // Astronomical Data Analysis Software an Systems XXIV (ADASS XXIV) : Vol 495. 2015 : 469.

[22] Cui C, He B, Yu C, et al. AstroCloud: A Distributed Cloud Computing and Application Platform for Astronomy[J]. arXiv preprint arXiv:1701.05641, 2017.

[23] He B, Cui C, Fan D, et al. AstroCloud, a Cyber-Infrastructure for Astronomy Research: Data Archiving and Quality Control[C] // Astronomical Data Analysis Software an Systems XXIV (ADASS XXIV) : Vol 495. 2015 : 483.

[24] Taylor M B. TOPCAT & STIL: starlink table/VOTable processing software[C] // Astronomical Data Analysis Software and Systems XIV : Vol 347. 2005 : 29.

[25] Robitaille T P, Tollerud E J, Greenfield P, et al. Astropy: A community Python package for astronomy[J]. Astronomy & Astrophysics, 2013, 558 : A33.

[26] Demleitner M, Plante R, Linde T, et al. IVOA Identifiers Version 2.0 (Recommendation)[EB/OL]. (2016-05-23) [2017-02-06]. http://www.ivoa.net/documents/IVOAIdentifiers/20160523/REC-Identifiers-2.0.pdf.

[27] Plante R, Stébé A, Benson K, et al. VODataService: a VOResource Schema Extension for Describing Collections and Services Version 1.1 (Recommendation)[EB/OL]. (2010-12-02) [2017-02-06]. http://ivoa.net/documents/VODataService/20101202/

REC-VODataService-1.1-20101202.pdf.

[28] Graham M, Morris D, Rixon G, et al. VOSpace Version 2.1 (Working Draft)[EB/OL]. (2016-04-20) [2017-02-06]. http://ivoa.net/documents/VOSpace/20160420/WD-VOSpace-2.1-20160420.pdf.

[29] Donovan A A, Kernighan B W. Go 程序设计语言（英文版）[M]. 北京 : 机械工业出版社, 2016.

[30] TIOBE. TIOBE Index for 2016[EB/OL]. (2016/12/31) [2017/01/01]. http://www.tiobe.com/tiobe-index/.

[31] Pengfei Y. GZIP、LZO、Zippy/Snappy 压缩算法应用场景小结 [EB/OL]. (2012/12/24) [2017/02/13]. http://www.cnblogs.com/panfeng412/archive/2012/12/24/applications-scenario-summary-of-compression-algorithms.html.

[32] Gilbert S, Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services[J]. Acm Sigact News, 2002, 33(2) : 51 – 59.

[33] Hanisch R, Greene G, Linde A, et al. Resource metadata for the Virtual Observatory[C] // Astronomical Data Analysis Software and Systems (ADASS) XIII : Vol 314. 2004 : 273.

[34] Matthew J G, ThomasAM M. 虚拟天文台: 天文学研究的工具与技术 [M]. [S.l.] : 北京: 中国科学技术出版社, 2010.

# 致谢

2013 年 9 月以来难忘的硕士学习生活即将结束，在此要对许多给过我热心指导帮助的老师同学表示衷心的感谢!

首先感谢我的指导教师—王华锋副教授，王老师是一个有着丰富经验、技术高超的老师，每次的交流总能得到新鲜的营养。

感谢我的企业导师—中国科学院国家天文台的刘继峰研究员，他经验丰富，也为我提供了不少有益的建议。

感谢北航大数据技术与应用 2013 班的班主任邹佳凝老师，邹老师一直在为我们班级提供着尽心尽力的服务，非常感谢。

感谢本文的评阅人张天柱老师和刘园老师，你们对论文都给了很好的建议。

感谢答辩秘书邓莹莹老师，您对我们的帮助非常重要。

感谢毕业答辩的几位评委老师，你们的点评给了我不少新的启发和思考，开拓了我的眼界。

感谢北航大数据技术与应用 2013 班的各位同学，与你们在学习、工作、生活上的交流是一件非常享受的事情，非常感谢你们的帮助。

感谢中国科学院国家天文台“中国虚拟天文台”团队的所有兄弟姐妹。我们是一个完美的团队，在本文的完成过程中得到了你们大量的帮助，也吸收了你们非常好的建议。

感谢天津大学计算机学院于策博士、肖健博士，对于系统研发过程中遇到的一些问题，你们给出了非常好的建议。

感谢中国科学院国家天文台 LAMOST 望远镜、兴隆 2.16 米望远镜，云南天文台丽江 2.4 米望远镜、抚仙湖太阳望远镜等望远镜运行团队，在系统的调试中得到了你们非常多的帮助。

本文的工作得到了国家自然科学基金—天文联合基金项目:“天文多节点海量数据归档的关键技术研究（U1531115)”的资助，在此也表示感谢。

最后感谢我的父母、我的妻子和儿子，有你们的支持，是我最大的动力和幸福。