中国科学院大学
University of Chinese Academy of Sciences

硕士学位论文

面向 21CMA 的分布式存储运维管理系统设计与实现

作者姓名：杨嘉宁

指导教师：崔辰州 研究员 中国科学院国家天文台

学位类别：理学硕士

学科专业：天文技术与方法

培养单位：中国科学院国家天文台

2024 年 6 月

Design and Implementation of Distributed Storage Operation and Maintenance Management System for 21 Centimeter Array

A thesis submitted to

University of Chinese Academy of Sciences

in partial fulfillment of the requirement

for the degree of

Master of Natural Science

in Astronomical Technology and Method

By

YANG Jianing

Supervisor: Professor CUI Chenzhou

National Astronomical Observatories, Chinese Academy of Sciences

June, 2024

中国科学院大学
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工作所取得的成果。承诺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享有著作权的研究成果，未在以往任何学位申请中全部或部分提交。对本论文所涉及的研究工作做出贡献的其他个人或集体，均已在文中以明确方式标明或致谢。本人完全意识到本声明的法律结果由本人承担。

作者签名：杨嘉宁

日　　期：2024年5月31日

中国科学院大学
学位论文授权使用声明

本人完全了解并同意遵守中国科学院大学有关收集、保存和使用学位论文的规定，即中国科学院大学有权按照学术研究公开原则和保护知识产权的原则，保留并向国家指定或中国科学院指定机构送交学位论文的电子版和印刷版文件，且电子版与印刷版内容应完全相同，允许该论文被检索、查阅和借阅，公布本学位论文的全部或部分内容，可以采用扫描、影印、缩印等复制手段以及其他法律许可的方式保存、汇编本学位论文。

涉密及延迟公开的学位论文在解密或延迟期后适用本声明。

作者签名：杨嘉宁　　导师签名：崔

日　　期：2024年5月31日　　日　　期：2024.5.31

摘 要

21 厘米阵列望远镜（21 CentiMeter Array，21CMA）是由中国科学院国家天文台主持建设的低频射电阵列，同时也是我国在平方公里阵列（Square Kilometre Array，SKA）低频波段的先导设备。为赋予 21CMA 脉冲星搜寻能力，国家天文台正在科技部 SKA 专项资助下对其进行升级和改造。升级后的 21CMA 每小时将产生数 TB 的数据。为解决海量数据接收及存储问题，21CMA 项目引入了基于 ARM 架构的低功耗并行存储设备。

21CMA 的数据存储系统将包括数十套低功耗并行存储设备，每套设备包含多个服务器节点和硬盘。在数十套低功耗并行存储设备上部署分布式文件系统将是一项复杂且庞大的任务，完全依靠手动部署不仅费时费力，也无法保证集群配置的一致性。另外，21CMA 位于新疆天山深处，位置偏僻，后期的运行和维护完全依靠人力非常困难，也不能保证及时性。

本文的研究目标包括：标准化存储设备的部署流程，实现系统配置的一致性，同时减少时间成本、降低出错概率；实现数据存储设备的故障恢复能力，解决系统的可靠性及可复用性需求；实现多集群存储设备统一监控能力，使系统在管理上更为便捷。本文的主要工作如下：

1. 针对 21CMA 数据存储设备部署流程复杂的问题，本文提出了一种基于 Ansible 的自动化部署模块解决方案。本文根据节点 IP 自动生成节点在分布式文件系统中的 ID，并根据用户对节点的分配自动完成 BeeGFS 的部署。本文还提供了一个用户友好的界面，方便用户对集群进行管理。该模块保证了集群的一致性，并提升了集群管理的便捷性。

2. 针对 21CMA 数据存储设备集群运行的稳定性需求和环境导致的运维不便，本文提出了一种改进的自动化故障恢复模块解决方案。BeeGFS 原有的故障恢复能力有限，一旦镜像组的两个节点都出现故障，数据将无法恢复。本文在原有节点类型的基础上增加了备用节点，并通过修改配置文件等一系列操作，实现了节点的自动替换和数据迁移。该模块使得 21CMA 数据存储设备能够在部分设备发生故障时保证系统的可用性和数据的完整性，同时减少了对人工干预的需求。

3. 针对 21CMA 数据存储设备的资源监测需求，本文结合 BeeGFS 的监控接口对其进行了扩展，实现了在更多维度上的资源监控。BeeGFS 原有的监控功能缺乏对集群的存储使用量以及集群存储设备硬件状态的监控。该模块通过远程调用，获取了每个存储节点的使用量和硬件使用状况，同时汇总了原有节点和第 2 点中提到的备用节点运行状态、集群读写速率、故障恢复日志、集群存储容量信息，直接向用户展示这些关键信息。

4. 针对 21CMA 数据存储设备多集群的特性，本文提出了一种多集群统一监控模块解决方案。本文将 BeeGFS 监控模块收集的监控数据分为能够展示节点运

行情况的关键数据和其他数据。该模块实时从多个集群的监控节点获取关键监控数据，并通过集中的页面展示在前端。这种方式不仅节省了查看每个集群监控页面的时间，还降低了漏掉关键信息的风险。

本文的研究成果是提出了一系列创新的天文数据存储设备管理方案，并构建了一套运维管理系统，集成了自动化部署及资源管理模块、自动化故障恢复模块、单个集群监控模块和统一监控服务模块，有效地解决了低功耗并行存储设备在一致性、可靠性和便捷性方面的不足。根据测试结果，部署模块能够在 0.4s 内完成对节点管理和软件部署请求的响应，保证了部署的一致性和可复现性，并且降低了部署的时间和人力成本。故障恢复模块在 BuddyMirror 模式下，能够在 50s 内完成系统数据初始化及启动动作，数据安全性得到了提升；在 Raid0 模式下，能够在 10s 内完成文件系统重构，减少了系统的故障时间。统一监控模块的数据库稳定性良好，能够满足 21CMA 多集群监控的需求。

如何科学地处理和存储海量天文数据已经成为天文学中的重要问题。采用分布式文件系统存储望远镜数据已经成为一种趋势。本文从 21CMA 望远镜的需求出发，在天文观测数据存储和管理方面提出创新的解决方案，展示了定制化存储系统的潜力和优势，为未来类似的低频阵列射电望远镜项目提供了宝贵的经验。

关键词： 21 厘米低频射电阵列；运维管理；监控可视化；分布式存储

Abstract

The 21 CentiMeter Array (21CMA) is a low-frequency radio array built under the auspices of the National Astronomical Observatory of the Chinese Academy of Sciences (NAOC), and is also China's pioneering equipment in the low-frequency band of the Square Kilometre Array (SKA). In order to give 21CMA pulsar search capability, the National Astronomical Observatory is upgrading and remodeling it under the special funding of the SKA from the Ministry of Science and Technology (MOST). The upgraded 21CMA will generate several terabytes of data per hour. To solve the problem of receiving and storing massive data, the 21CMA project has introduced a low-power parallel storage device based on the ARM architecture.

The data storage system of 21CMA will include tens of sets of low-power parallel storage devices, each of which contains multiple server nodes and hard disks. Deploying a distributed file system on tens of low-power parallel storage devices is a complex and huge task, and relying on manual deployment is not only time-consuming and labor-intensive, but also fails to ensure the consistency of the cluster configuration. In addition, 21CMA is located deep in the Tianshan Mountains of Xinjiang, which is a remote location, and it is very difficult to rely on manpower for operation and maintenance in the later stage, and it is also impossible to guarantee the timeliness.

The research objectives of this paper include: standardizing the deployment process of storage devices to achieve consistency in system configuration, while reducing the time cost and error probability; realizing the failure recovery capability of data storage devices to address the reliability and reusability needs of the system; and realizing the unified monitoring capability of multicluster storage devices to make the system more convenient in management. The main work of this paper is as follows:

1. In response to the complexity of the deployment process of 21CMA data storage devices, this paper proposes an automated deployment module solution based on Ansible. This paper automatically generates the node's ID in the distributed file system based on the node's IP, and automatically completes the deployment of BeeGFS based on the user's assignment of the node. This paper also provides a user-friendly interface to facilitate the management of the cluster. This module ensures cluster consistency and enhances the ease of cluster management.

2. In response to the stability requirements of 21CMA data storage device cluster operation and the inconvenience of operation and maintenance caused by the environment, this paper proposes an improved automated fault recovery module solution. the original fault recovery capability of BeeGFS is limited, and once both nodes of the mirrored group fail, the data will not be recovered. This paper adds a standby node to the

original node type, and realizes automatic node replacement and data migration through a series of operations such as modifying configuration files. This module enables the 21CMA data storage device to ensure system availability and data integrity in the event of partial device failure, and at the same time to reduce the need for manual intervention.

3. For the resource monitoring requirements of 21CMA data storage devices, this paper extends the monitoring interface of BeeGFS to realize resource monitoring in more dimensions. the original monitoring function of BeeGFS lacks the monitoring of the storage usage of the cluster as well as the hardware status of cluster storage devices. The module obtains the usage and hardware usage status of each storage node through remote invocation, and also summarizes the operation status of the original nodes and the standby nodes mentioned in point 2, the cluster I/O rate, the fault recovery log, and the cluster storage capacity information, and directly displays this critical information to the user.

4. Aiming at the multi-cluster characteristics of 21CMA data storage devices, this paper proposes a multi-cluster unified monitoring module solution. In this paper, the monitoring data collected by the BeeGFS monitoring module is categorized into key data that can show the operation of nodes and other data. The module obtains key monitoring data from the monitoring nodes of multiple clusters in real time and displays them on the front-end through a centralized page. This approach not only saves the time to view the monitoring page of each cluster, but also reduces the risk of missing critical information.

The research result of this paper is to propose a series of innovative management schemes for astronomical data storage devices and construct an operation and maintenance management system, which integrates the automated deployment and resource management module, automated fault recovery module, individual cluster monitoring module, and unified monitoring service module, and effectively solves the shortcomings of low-power parallel storage devices in terms of consistency, reliability, and convenience. According to the test results, the deployment module is able to complete the response to node management and software deployment requests within 0.4s, which ensures the consistency and reproducibility of the deployment, and reduces the time and labor cost of deployment. The failure recovery module is able to complete the system data initialization and startup action within 50s in BuddyMirror mode, and the data security is improved; in Raid0 mode, it is able to complete the file system reconfiguration within 10s, which reduces the system failure time. The database stability of the unified monitoring module is good enough to meet the needs of 21CMA's multi-cluster monitoring.

How to scientifically process and store massive astronomical data has become an important issue in astronomy. It has become a trend to use distributed file system to

store telescope data. Starting from the needs of the 21CMA telescope, this paper proposes innovative solutions in the storage and management of astronomical observation data, demonstrates the potential and advantages of customized storage systems in improving the reliability and efficiency of large-scale data storage, and provides valuable experience for similar low-frequency array radio telescope projects in the future.

Key Words: 21 CentiMeter Array; Operations and maintenance management; Monitoring and visualization; Distributed storage system

目 录

第 1 章 绪论 ····· 1
1.1 背景 ····· 1
1.1.1 SKA 及脉冲星搜寻预研 ····· 1
1.1.2 升级 21CMA 所面临的数据接收与存储问题 ····· 2
1.1.3 射电望远镜数据存储案例 ····· 3
1.2 低功耗分布式存储系统及终端设备 ····· 5
1.3 主要研究内容 ····· 6
1.4 文章结构 ····· 7
第 2 章 存储技术分类及分布式文件系统对比分析 ····· 9
2.1 存储技术分类 ····· 9
2.1.1 直连存储及网络存储技术 ····· 9
2.1.2 不同的数据访问方式 ····· 10
2.1.3 不同的存储架构 ····· 10
2.2 流行分布式文件系统分析及对比 ····· 11
2.2.1 GlusterFS ····· 11
2.2.2 Ceph ····· 12
2.2.3 Lustre ····· 13
2.2.4 BeeGFS ····· 13
2.2.5 GlusterFS、Ceph、Lustre、BeeGFS 对比 ····· 14
2.3 小结 ····· 15
第 3 章 21CMA 数据存储集群运维管理方法研究 ····· 17
3.1 21CMA 数据存储集群需求分析 ····· 17
3.2 21CMA 存储设备运维管理系统功能需求 ····· 18
3.3 BeeGFS 应用及功能分析 ····· 19
3.3.1 BeeGFS 部署流程 ····· 19
3.3.2 BeeGFS 文件条带化原理 ····· 19
3.3.3 BeeGFS 控制工具 ····· 20
3.3.4 BeeGFS 监控服务 ····· 21
3.4 系统架构设计 ····· 22
3.5 模块功能设计 ····· 23
3.6 模块部署方案 ····· 24
3.7 小结 ····· 25

第 4 章 21CMA 数据存储设备运维管理系统实现 ····· 27
4.1 系统实现技术需求及选择 ····· 27
4.1.1 网站实现技术需求及选择 ····· 27
4.1.2 远程配置管理实现技术需求及选择 ····· 28
4.1.3 监控服务数据抽取技术选择 ····· 28
4.2 数据库设计与实现 ····· 29
4.2.1 单个集群运维管理系统数据库设计与实现 ····· 29
4.2.2 统一监控网站数据库设计与实现 ····· 30
4.3 集群部署及软硬件管理模块实现 ····· 32
4.3.1 节点管理 ····· 32
4.3.2 密钥生成及分发 ····· 33
4.3.3 磁盘扫描及管理 ····· 34
4.3.4 角色管理与部署 ····· 36
4.3.5 伙伴组管理 ····· 38
4.3.6 存储池管理 ····· 39
4.4 自动化故障恢复模块实现 ····· 40
4.4.1 故障检查及恢复流程 ····· 40
4.4.2 节点替换及数据恢复功能 ····· 41
4.5 单个集群监控模块实现 ····· 42
4.5.1 集群关键数据提取 ····· 42
4.5.2 集群数据可视化 ····· 43
4.6 统一监控服务模块实现 ····· 44
4.7 小结 ····· 46
第 5 章 系统应用及测试 ····· 47
5.1 系统应用 ····· 47
5.1.1 单个集群运维管理系统部署及应用 ····· 47
5.1.2 统一监控网站部署及应用 ····· 47
5.2 系统测试 ····· 48
5.2.1 测试环境 ····· 48
5.2.2 部署及软硬件管理模块测试 ····· 48
5.2.3 故障恢复模块测试 ····· 49
5.2.4 统一监控模块测试 ····· 50
5.3 小结 ····· 50
第 6 章 总结 ····· 51
6.1 结论与创新点 ····· 51
6.2 不足与展望 ····· 51

参考文献 ····· 53
致谢 ····· 55
作者简历及攻读学位期间发表的学术论文与其他相关学术成果 · 57

图目录

图 1-1 SKA 现状 …… 2
图 1-2 21CMA 望远镜阵列 …… 3
图 1-3 基于 ARM 架构的存储系统 …… 6
图 2-1 BeeGFS 文件系统基本架构 …… 14
图 3-1 故障检查及恢复流程图 …… 20
图 3-2 beegfs_mon 数据库结构 …… 21
图 3-3 21CMA 存储运维管理系统架构图 …… 23
图 3-4 21CMA 存储系统架构图 …… 25
图 4-1 集群数据库 ER 图 …… 31
图 4-2 统一监控数据库 ER 图 …… 31
图 4-3 节点展示及管理页面 …… 32
图 4-4 密钥生成及分发页面 …… 33
图 4-5 磁盘展示及管理页面 …… 35
图 4-6 角色展示及管理页面 …… 36
图 4-7 角色部署基本流程 …… 37
图 4-8 伙伴组展示及管理页面 …… 38
图 4-9 存储池展示及管理页面 …… 39
图 4-10 故障检查及恢复流程图 …… 41
图 4-11 资源展示页面 …… 44
图 4-12 存储设备监控数据 …… 45
图 4-13 存储设备的统一监控及可视化 …… 45
图 5-1 统一监控服务数据库查询耗时 …… 50

表目录

表 1-1 存储设备硬件信息 …… 5
表 5-1 软硬件操作时间及成功率 …… 49
表 5-2 单节点部署操作时间及成功率 …… 49
表 5-3 故障恢复时间及成功率 …… 49

第 1 章　绪论

1.1　背景

天文学起源于人类对夜空的好奇心和敬畏之情，是历史上最早的科学领域之一。天文学研究内容广泛，包括恒星、行星、卫星等天体，以及彗星、流星、超新星等各类天体现象，还有宇宙的整体结构与演变过程。天文观测是天文学的重要组成部分，是获取天文学研究数据的主要途径。通过观测，天文学家可以收集到各类天体的光谱、亮度、位置、运动状态等信息。这些信息是进行天文学研究的基础。通过对这些信息的分析和解释，天文学家可以了解天体的性质和宇宙的演化规律。

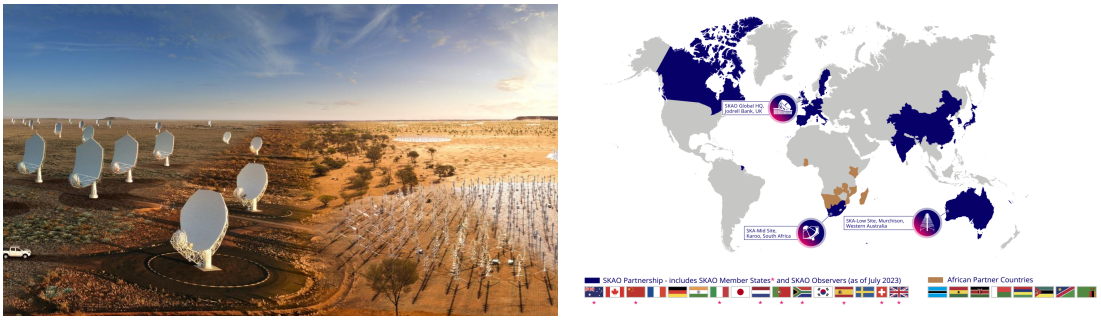
大约 400 年前，伽利略改进了望远镜并首次将其用于天文观测，使得人类能够超越肉眼的限制，更深入地探索宇宙。这一阶段的发展为哥白尼的日心说提供了实证，促进了科学革命的发展。1932 年卡尔 · 央斯基（Karl Guthe Jansky）使用自己建造的天线观测到了来自银河系中心的无线电波，这一发现标志着无线电天文学诞生 (Singh, 2005)。射电天文学的重要性在于它打开了一扇新的窗口，让天文学家能够观测到那些在可见光波段无法看到的天体和现象。射电望远镜能够探测到遥远的星系、星云、脉冲星以及其他宇宙中的极端环境，这些观测对于理解宇宙的大尺度结构和复杂过程至关重要。射电波段的观测还可以穿透星际尘埃云，揭示那些被尘埃遮蔽的天体，如恒星形成区域和星系中心的超大质量黑洞。

随着技术的进步，射电天文学已经发展出多种观测手段，如甚长基线干涉测量（Very Long Baseline Interferometry，VLBI）(Schuh 等, 2012)，这种技术可以极大地提高测量的角分辨率，使得天文学家能够精确地测量天体的位置和运动。此外，射电阵列望远镜如美国的甚大阵列（Very Large Array，VLA）(Napier 等, 1983) 和全球的阿塔卡马大型毫米/亚毫米阵列（Atacama Large Millimeter Array，ALMA）(Brown 等, 2004) 等，能够观测到宇宙中的冷气体和尘埃，这些是恒星和行星形成的重要成分。然而，随着观测技术的不断发展，观测数据量也呈现出爆炸性增长的趋势，这给天文数据接收、存储、处理、分析都带来了新的挑战。

1.1.1　SKA 及脉冲星搜寻预研

平方公里阵列（Square Kilometre Array，SKA），如图 1-1(a) 所示，是国际天文界计划建造的世界上最大的综合孔径望远镜，建成后将会是国际天文学界的一个里程碑 (Schaubert 等, 2003)。SKA 的主要科学目标包括探究宇宙起源、探索星系演化、研究恒星、星系、引力波、射电爆、脉冲星等众多天体和天体物理现象 (Carilli 等, 2004)。该项目自 1993 年启动，在过去十年间，包括中国在内的 20 个国家的大约 100 个组织参与了望远镜的设计与开发，如图 1-1(b) 所示。

SKA 涵盖了从低频到高频的射电波段，总面积超过一平方公里，远大于现有



(a) SKA 碟形阵列和 SKA 天线 **(b) SKA 成员国**

(a) SKA dish arrays (left) and antenna arrays (right) **(b) Member countries of SKA**

图 1-1 SKA 现状

Figure 1-1 Status of SKA

任何射电望远镜，将产生前所未有的数据量。SKA 的建设分为 SKA-I 和 SKA-II 两个阶段进行。第一阶段正式运行后每年需要归档的科学数据约为 710PB(Guo 等, 2023)，SKA 的数据量给数据存储及计算都带来了巨大的挑战。

脉冲星搜寻是 SKA-I 阶段的首要科学目标之一。国外的许多射电望远镜都开展了脉冲星搜寻研究，澳大利亚的 ASKAP（Australian Square Kilometre Array Pathfinder）、澳大利亚的默奇森宽视场阵列（Murchison Widefield Array），南非的 MeerKAT，以及荷兰的 LOFAR（Low-Frequency Array）和 ATA（Allen Telescope Array）等先驱阵列和探路者项目都已取得一定的成果。

1.1.2 升级 21CMA 所面临的数据接收与存储问题

21 厘米阵列望远镜（21 CentiMeter Array，21CMA）项目始于 2004 年，是由中国科学院国家天文台主持建设的低频射电阵列，主要科学目标是探测宇宙黎明和第一代发光天体时期中性氢原子的 21 厘米信号，从而揭示宇宙从黑暗走向光明的历史，是世界上最早开展搜寻宇宙第一缕曙光的大型射电望远镜阵列 (徐怡冬 等, 2020)。21CMA 位于新疆乌拉斯台站，由 81 个子阵组成，共计 10287 个天线, 如图 1-2 所示。21CMA 的优点是它的大视场和长基线。大视场意味着它可以一次观测到天空的大片区域，而长基线则意味着它具有很高的分辨率，可以观测到非常微弱的射电源。21CMA 的有效工作频率为 50~250MHz，是目前中国唯一可以与未来 SKA1-low 衔接的探路者，也是实现低频脉冲星搜寻的优秀平台 (武向平, 2019; Zheng 等, 2016; Jongerius 等, 2014)，但 21CMA 目前尚不具备脉冲星观测能力。

为此，科技部启动了 SKA 预研项目，通过对 21CMA 技术升级使其具备脉冲星观测能力。升级后的 21CMA，首先通过模拟设备实现子阵波束合成，然后采用专用终端进行波束合成和相干消色散。近年开展的大规模低频脉冲星巡天，MWA 的非相干波束脉冲星巡天的数据量为 28TB/h(Xue 等, 2017)，LOFAR 的 Tied-Array All-Sky Survey 的数据量为 17TB/h(Sanidas 等, 2019)。21CMA 升级完成后，其脉冲星巡天产生的数据量远超其他低频脉冲星巡天项目，若 81 个子

图 1-2　21CMA 望远镜阵列

Figure 1-2　21 CentiMeter Array

阵同时运行，将产生海量的数据。在 SKA 专项预研阶段，为了挖掘更多的科学价值，需要尽可能保留原始数据，这需要存储设备具备更高的存储容量和数据接收速率，对 21CMA 的存储设备的成本、性能及扩展性提出了更高的要求。

1.1.3　射电望远镜数据存储案例

随着望远镜技术进步和观测能力增强，数据量的增长在全球范围内的天文研究中普遍存在。早期望远镜的每年数据产量可能仅限于几百 MB，而 SKA 等现代望远镜每年预计将生成 TB、PB 乃至 EB 级别的数据量。数据量的增加对存储系统提出了巨大挑战，传统的存储模式在经济成本、传输速率、存储容量方面都不再能满足天文望远镜数据存储的需求。

随着数据量的不断扩大，天文数据的存储方式在不断演变。早期数据存储在磁带或硬盘上，而现在更多地采用高性能计算环境中的大型存储阵列，或转向云存储方案，以增强数据存取效率和可靠性。以下从单口径到望远镜阵列，列举了国内外部分射电望远镜以实际需求为基础定制或部署的数据存储系统，分布式存储技术已逐步成为应对海量数据的优选方案。

1. GBT(Green Bank Telescope) 是一台 100m×110m 的椭圆形天线，观测频率 0.3~115GHz，数据量达 24GB/s。存储方案为 32 个计算节点和 4 个存储节点，每个存储节点包括 36 个 6TB 的硬盘，配置为 3 组 RAID 6 阵列。每个 RAID 阵列的可用存储容量为 50TB，每个节点的容量为 150TB(MacMahon 等, 2018)。

2. Parkes 是一台 64m 的单口径望远镜，主要观测频率 1.23~1.53 GHz，采用与 GBT 相同的数据记录系统。存储方案为 27 个计算节点和 4 个存储节点，每个存储节点配备 36 个 6TB 的硬盘，配置为 3 组 RAID5 阵列。每个 RAID 卷的可用存储容量为 55TB，每个节点可用存储容量为 165TB(Price 等, 2018)。

3. 新疆天文台南山观测站 25m 射电望远镜 (NanShan Radio Telescope, NSRT) 经改造后口径扩大到 26m，观测频率 320~2200MHz(刘奇 等, 2019)。存储系统以

Lustre 为基础，由两个主备模式的元数据服务器和三台目标存储节点共同构成，总计提供 100TB 的存储空间，解决了观测数据及次生数据的在线存储及数据安全问题 (张海龙 等, 2018a)。

4. 新疆奇台 110m 口径射电望远镜 (QiTai radio Telescope, QTT) 观测频率 150MHz~115GHz。数据存储和备份系统包括高速网络连接的数据中心存储（Tier0）、本地存储（Tier1）以及分布式存储（Tier2）(张海龙 等, 2018b; 王娜, 2014)。QTT 借助 BeeGFS 构建更高效的数据存储后端，提高 I/O 性能，实现高性能和可扩展性 (张萌 等, 2021)。

5. ASKAP (The Australian SKA Pathfinder) 由 36 个直径 12m 的反射面天线组成，观测频率 30~950GHz，每秒数据量可达 2.5GB。Pawsey 超级计算中心通过 Lustre 共享文件系统为 ASKAP 提供了 1PB 的临时容量 (Guzman 等, 2016)。最终数据存储在 CASDA(The CSIRO ASKAP Science Data Archive) 中，CASDA 存储硬件包括磁带和 MAID (Massive Array of Idle Disks) 两种，前者用于保存大文件，后者用于保存小文件。

6. LOFAR (Low Frequency Array) 包括 10~90MHz 的低频阵和 110~250MHz 的高频阵，每年产生约 5PB 的科学数据 (Renting 等, 2012; Holties 等)。数据存储在 LOFAR 长期存档中，其基于 Astro-WISE 架构，构建了一个分级存储体系：Tier 0 负责数据采集与初步处理；Tier 1 提供数据服务器或基于网格的存储设施；Tier 2 面向外部用户，同时也服务于虚拟天文台等天文学社区 (Belikov 等, 2011)。此外。意大利 Trieste 天文台还使用 BeeGFS 构建了高性能计算平台用于 LOFAR 数据处理和数据分析 (Bertocco 等, 2020)。

7. MWA (Murchison Widefield Array) 有 16 个子阵，由 2048 个对数天线组成，观测频率 80~300MHz，用于脉冲星观测。数据接收由 16 台服务器实现，每台服务器拥有 2 个 1.44TB 的 RAID 存储，之后数据归档到澳大利亚珀斯超级计算中心 (Ord 等, 2019)。数据归档包括在线存储、长期存储和镜像归档。在线存储有两个节点，每个节点连接 24 个 SAS 硬盘驱动器，总容量为 80TB。长期归档包括 PBStore 和 Fornax，其中 PBStore 通过高速光纤定期从在线存档获取数据，Fornax 是一个由 96 个计算节点组成的 GPU 计算集群，总计 672TB 本地存储。镜像存档位于各个国际站点，用于实现冗余和本地访问 (Wu 等, 2013)。

8. HERA(Hydrogen Epoch of Reionization Array) 是一个由 350 个 14 米抛物面型天线组成的干涉仪，观测频率 50~250MHz(DeBoer 等, 2017)。HERA 的主要长期存储实例位于 NRAO(National Radio Astronomy Observatory) 位于新墨西哥州索科罗的设施中。NRAO 的处理集群配备了 14 个计算节点，每个节点都有 8 核处理器和 256GB 的内存。这些计算节点通过 40Gbit Infiniband 网络连接到分布式并行文件系统 Lustre，以进行数据访问和处理 (Berkhout 等, 2024)。

8. MeerKAT 由 64 个碟形天线组成。MeerKAT 天文台的数据处理和存储系统包括两个主要部分：Filterbanking Beamformer User Supplied Equipment (FBFUSE) 和 Accelerated Pulsar Search User Supplied Equipment(APSUSE)。其中 APSUSE 是

一个高性能集群，可以捕获并存储 FBFUSE 生成的高达 280Gb/s 的数据。FBFUSE 和 APSUSE 共享一个 BeeGFS 分布式文件系统上的 3.5PB 的公共文件存储，该系统能够产生高达 50GB/s 的 IO 速度 (Padmanabh 等, 2023)。

1.2　低功耗分布式存储系统及终端设备

分布式存储已经成为解决望远镜数据接收和存储的重要方案之一。随着存储速率和存储容量需求的不断增长，数据存储的规模也在不断扩大，数据存储系统的能耗也逐渐成为了关注焦点之一。如何减少数据存储和处理的能耗也成为现代天文数据存储策略研究的一部分。

目前射电望远镜数据存储均采用传统的分布式文件系统。传统的分布式文件系统以 x86 架构服务器为平台，在其上部署分布式文件系统。x86 架构在数据密集型工作中应用广泛，并经过了长期的改进和检验，具有成熟、稳定、速率高的优点。然而，x86 架构的处理器通常有较高的能耗，在处理密集型任务时会产生大量热量，这增加了散热和电力成本。在大规模部署和应用时，使用 x86 架构服务器会导致在成本、功耗等方面付出较高的代价。

为解决这一问题，韩军等提出了低功耗并行存储系统作为解决方案 (韩军 等, 2023)。该设备包括多个节点及网络设备和电源设备，分别用于提供分布式存储服务、终端设备的访问以及提供网络通信连接服务以及电源提供服务。其中，每个节点采用基于 ARM 架构的芯片作为其核心计算单元，ARM 架构的芯片具有体积小、功耗低的优点。它解决了传统以 x86 架构为核心的分布式存储系统所面临的功耗高和占用物理空间大的问题。

低功耗并行存储系统的硬件信息如表 1-1 所示。在 21CMA 应用场景中，低功耗并行存储设备使用树莓派作为节点。树莓派可以运行多种操作系统，包括官方提供的 Raspberry Pi OS（也称为 Raspbian），以及其他 Linux 发行版如 Ubuntu。Raspberry Pi OS 和 Ubuntu 都是基于 Debian 的 Linux 发行版，因此它们共享相同的命令行接口和基本命令集。无论是在 Raspberry Pi OS 还是 Ubuntu 系统中，用户可以使用一致的命令语法操作系统和管理任务。

该技术是 SKA 专项预研探索的技术方案之一，目前已经实现了 GB/s 级别的写入速度，可以满足 21CMA 子阵的数据存储速率需求。

表 1-1　存储设备硬件信息

Table 1-1　Hardware information of a storage device

硬件	配置信息
服务节点	树莓派 4B
硬件	20TB SATA 机械硬盘
交换机	H3C 万兆交换机

1.3 主要研究内容

由于数据量庞大，21CMA 数据存储系统将包括数十套低功耗并行存储设备，前期部署较为复杂。并且，21CMA 位于偏远地区，存储集群的后期运行和维护将会非常困难。尽管低功耗分布式存储系统及终端设备具备高存储速率、高存储容量、低功耗的优点，但是实现 21CMA 数据稳定接收和存储还是远远不够的。要高效地使用低功耗分布式存储系统及终端设备并确保其稳定地运行，还需要对其在并行文件系统适配度、软件能力、硬件能力三个方面进一步提升。如图 1-3 所示，本文的研究目标是提升低功耗分布式存储系统及终端设备的软件能力。

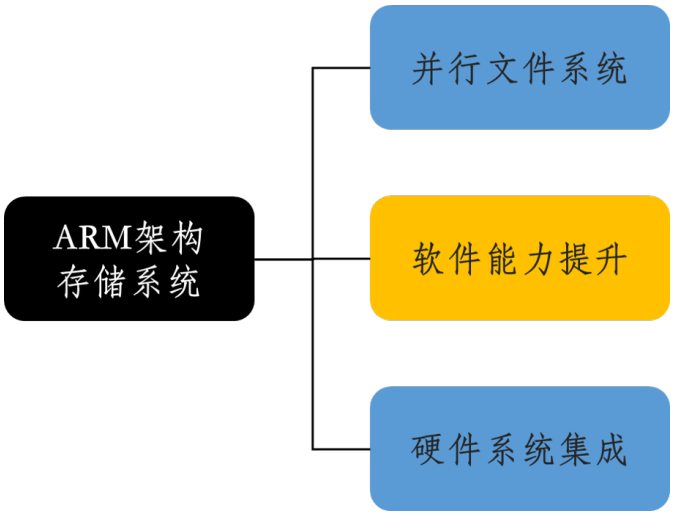


图 1-3 基于 ARM 架构的存储系统

Figure 1-3 Storage system based on ARM architecture

本文针对低功耗分布式存储系统及终端设备在软件方面中存在的问题，提出一套运维管理方法，并对该方法进行实现、应用、评估，最终构建统一的运维管理系统，实现对 21CMA 数据存储设备的部署管理、运行管理、设备维护和监控。本文的主要研究内容如下：

1. 对目前 21CMA 数据接收和存储的需求进行调研，分析低功耗分布式存储系统及终端设备存在的不足。针对调研和分析过程中提出的问题，围绕如何满足 21CMA 数据接收和存储的功能需求、提升低功耗分布式存储系统及终端设备的使用体验，设计面向 21CMA 数据存储设备的运维管理系统，提出系统架构和模块化的解决方案。

2. 根据解决方案，围绕技术需求选择合适的框架及工具。通过对部署流程的分析，并结合配置管理工具，实现部署流程自动化；通过对 BeeGFS 高可用实现的分析，集合远程调用和新添加的备用节点，实现自动化的故障恢复；通过扩展 BeeGFS 的监控接口，实现多维度集群资源监控；通过汇总关键数据，实现多集群统一监控。

3. 根据 21CMA 数据存储的应用场景，设计系统测试方案，包括对部署响应时间及成功率的测试、对在两种不同的条带模式下故障恢复能力的测试、对监

控服务的数据库在长时间运行后的稳定性测试。通过对系统的实际应用和测试，评估该运维管理系统，并提出改进方案。

1.4　文章结构

本文分为六大章节：

第一章为绪论，介绍了课题的的背景，望远镜数据存储的现状以及 21CMA 分布数据存储系统的基本需求，说明了研究目的及研究内容。

第二章为存储技术介绍及分布式存储方案分析，介绍现有存储技术，并对流行的分布式文件系统做了分析，阐明为何选择 BeeGFS 作为低功耗并行存储设备的存储方案。

第三章为 21CMA 数据存储设备运维管理方法研究，提炼了 21CMA 数据存储的需求，分析了低功耗并行存储设备和其采用的分布式文件系统 BeeGFS 的不足之处。并在此基础上，对 21CMA 数据存储的运维管理的架构、功能模块进行了概要设计。

第四章为系统的详细设计与实现，分析了系统的技术需求并在此基础上选择了实现工具，设计并实现了系统数据库，最后详细展开了如何对 BeeGFS 进行扩展和定制化。

第五章为系统应用及测试，介绍了系统的使用方法，并根据实际应用场景设计了一系列测试方案并进行了测试。

第六章为总结与展望，对本文内容做了总结，并阐述了以后可以改进和扩展的内容。

第 2 章　存储技术分类及分布式文件系统对比分析

选择存储方案时，需要综合考虑存储系统的网络连接方式、数据访问方式、性能、可扩展性、数据保护和冗余、运维管理成本等因素。

21CMA 观测过程中产生的数据量非常庞大，每小时可达 TB 级。为了尽量保留下来原始数据，在后期挖掘更多的科学价值，21CMA 数据存储设备需要具备高存储容量和高数据传输速率。为了确保 21CMA 数据的安全性和持久性，21CMA 数据存储和访问系统需要有数据备份、冗余和容错机制。

本章将介绍几种存储技术分类，并对几种流行的分布式文件系统进行分析和对比，选择适用于 21CMA 数据存储的分布式存储架构。

2.1　存储技术分类

本节将从网络连接方式、数据访问方式、存储架构三个角度对存储技术进行分类和介绍。

2.1.1　直连存储及网络存储技术

存储系统按网络连接性可分为直连式存储（Direct-Attached Storage，DAS）、存储区域网络（Storage Area Network，SAN）和网络附加存储（Network-Attached Storage，NAS）。

DAS 将存储设备通过小型计算机接口或光线通道与主机相连，其数据读写、备份和恢复等性能依赖服务器 CPU 和 IO 等资源。随着数据量的增长，需要对存储设备和主机进行升级，但由于物理连接限制，其扩展能力有限。解决这一问题的方式是将存储器从应用服务器中分离出来，进行集中管理，即存储网络（Storage Networks）。

实现存储网络有两种方式：SAN 和 NAS。SAN 通过光纤通道连接存储设备与主机，支持高带宽和远距离传输，但成本随节点增加而显著提高，网络设备的成本显著上升。其可扩展能力在大规模部署时受限。NAS 将存储设备通过标准的网络拓扑结构添加到服务器或服务器集群，支持跨平台文件共享，使得主机的文件系统不仅限于本地文件系统，还可以连接基于局域网的共享文件系统，满足了灵活增加存储容量的需求，但在数据备份和恢复时，除需要正常用户数据交互传输外，还必须处理备份操作的 IO 请求，带宽消耗较大。二者的关键区别在于文件管理系统位置。SAN 为每个服务器提供独立的文件管理；而 NAS 则是每个应用服务器通过网络共享协议共享一个文件系统。此外，从数据传输的角度看，SAN 和 DAS 传送数据块，不依赖标准文件共享协议，NAS 则基于文件传输，使用标准协议。

2.1.2 不同的数据访问方式

存储技术按数据访问方式分类可分为三种：块存储、文件存储、对象存储。本小节从组织形式、访问方式、优缺点三个角度对三种存储技术进行介绍。

块存储将数据分割成固定大小的块，每个块独立寻址，类似于硬盘驱动器的扇区。这种方法提供高性能和低延迟的访问，适合需要快速读写操作的应用，如数据库和关键业务应用程序。块存储易于扩展，添加新的存储块可以无缝扩充系统容量。

文件存储以文件和文件夹的层次结构来组织数据，用户通过文件路径访问数据。这种方式的优势在于直观和易于管理，支持跨平台文件共享，适合文档存储和共享应用场景。文件存储系统中的并行技术可以提升访问效率，适用于需要共享和频繁访问文件的环境。

对象存储采用去中心化和扁平化的结构来存储数据。在对象存储系统中，对象是最小的存储单元，每个对象包含数据和元数据，对象之间相互独立。这种方式支持海量数据的存储和访问，具有良好的可扩展性和灵活性。对象存储特别适合云存储环境，可以实现全球范围内的数据共享和访问。对象存储系统结合了 SAN 的高速直接访问物理磁盘的特性以及 NAS 的数据共享特性，从而继承了这两种存储技术的优势。

这三种存储方法针对不同的需求和场景，提供了各自的解决方案。块存储适用于性能敏感型应用，文件存储适合共享和管理文件，而对象存储则适应于大规模数据的存储和分布式访问。随着数据量的增长和应用需求的多样化，选择合适的存储方法对于提高数据管理效率和降低成本具有重要意义。

2.1.3 不同的存储架构

存储技术按照架构可以分为三种：传统存储、软件定义存储、云存储。

传统存储通常指的是物理存储设备，如硬盘驱动器（HDDs）和固态驱动器（SSDs），以及其组成的阵列。这些设备可以通过 DAS、SAN、NAS 提供数据存储。传统存储技术经过多年的发展，相对稳定可靠。

软件定义存储（Software-Defined Storage, SDS）是一种日益流行的存储架构，它从物理存储硬件中抽象出存储管理和服务 (Carlson 等, 2014)。SDS 通常使用标准的服务器硬件，并通过软件来提供和管理存储资源。因为 SDS 的存储服务由软件定义，可以快速适应变化的需求，集中化的管理界面简化了存储配置和操作。SDS 具有灵活、简洁、高效的优点。此外，由于使用商品硬件，SDS 还具有低成本的优点。

云存储是由云服务提供商提供的存储资源，可以通过互联网进行访问。这些存储资源可以提供块存储、文件存储或对象存储。云存储可以根据需求增减资源，非常灵活。并且，由于云服务提供商通常会在多个数据中心复制数据，因此具有较高的可用性。

SDS 通过将存储软件与硬件分离，消除了软件对专有硬件的依赖性，这是近年来出现的概念，并且涌现了多种实现形式，如分布式文件系统（GlusterFS、Lustre、BeeGFS）、分布式对象存储（MINIO）、分布式块存储（Sheepdog）和统一存储（Ceph）等。其中，BeeGFS 是近些年最为热门的分布式并行存储系统之一，是 I/O 密集型工作的优质存储解决方案，在高性能计算社区广受欢迎，近些年在天文数据计算和高速存储领域也被广泛采用。

2.2　流行分布式文件系统分析及对比

综合考虑传统存储、软件定义存储的优缺点以及 21CMA 数据存储和访问的需要，21CMA 最终采用分布式文件系统来实现存储系统。本节从架构、数据分布和数据冗余算法、适用场景角度对流行的分布式文件系统 GlusterFS(Boyer 等, 2012)、Ceph(Weil 等, 2006)、Lustre(Braam, 2019)、BeeGFS(Abramson 等, 2020) 进行介绍。

2.2.1　GlusterFS

GlusterFS 是一种可扩展的网络文件系统。GlusterFS 架构中没有元数据服务器组件，其主要组件只包括存储服务器（Brick Server）和客户端。GlusterFS 还包括一个中央管理守护进程，称为 Glusterd，它负责配置和维护整个文件系统。

块（Brick）和卷（Volume）是 GlusterFS 中的两个核心概念。块是存储的基本单位，它代表了连接到网络的、挂载在服务器上的一个目录。多个块可以组合形成卷。卷之间通过网络连接，跨越多台服务器。客户端可以通过本地文件系统挂载的方式访问卷。卷有以下五种：

（1）Distributed Gluster Volume：分布式卷，文件通过 hash 算法随机分布到由块组成的卷上。文件只能存在分布式卷中某一块上，而不能存储在多个块中。

（2）Replicated Gluster Volume: 复制卷（类似 RAID1），副本数必须等于卷中块所包含的存储服务器数，可用性高。

（3）Distributed Replicated Glusterfs Volume: 分布式复制卷，文件分布在块的复制集中，块数必须是副本数的倍数。

（4）Dispersed Glusterfs Volume：纠错卷，基于纠错码，跨卷中的多个块对文件的编码数据进行条带化，并添加了一些冗余。

（5）Distributed Dispersed Glusterfs Volume: 分布式纠错卷，卷中块所包含的存储服务器数必须是副本的倍数，兼顾分布式和复制式的功能。

GlusterFS 使用分布式哈希表（Distributed Hash Table，DHT）将请求映射到包含所需文件或目录的正确块；使用自动文件复制（File Replication，AFR）来跟踪文件操作，实现跨块复制数据，如果卷是复制卷，它会复制该请求并将其传递给副本的协议客户端转换器。纠错卷和分布式纠错卷还采用了纠删码（Erasure Code）算法。每个块仅存储每个数据块的一部分，其中某些块包括了奇偶校验或冗余块，通过数学变换可以用于恢复存储在另其他块上的内容。

GlusterFS 提供的多种卷能够满足不同场景下的存储需求。分布式卷允许数据在多个设备间自动分布，以便进行横向扩展。复制卷和纠错卷确保了数据的冗余存储，提高了系统的容错能力。条带功能则针对大文件的存储和访问进行了优化。此外，GlusterFS 支持多种协议，包括 NFS、SMB/CIFS 和 GlusterFS 本身的访问协议，使其能够与各种环境兼容，并且提供了命令行界面和 Web 界面，使得系统的管理既简单又直观。

2.2.2 Ceph

Ceph 是一种高度可扩展的分布式存储系统，提供对象、块和文件存储能力，设计目标是满足各种规模和环境的多样化存储需求。Ceph 的架构是去中心化的，它由监视器（Ceph Monitor，MON）、管理器守护进程（Manager，MGR）、对象存储守护进程（Object Storage Daemon，OSD）、元数据服务（Ceph Metadata Server，MDS）四个主要组件构成。MON 维护着集群的映射。OSD 负责存储数据、处理数据复制、恢复、再平衡，并通过检查其他 OSD 守护进程的心跳将监视信息提供给监视器和管理器。MGR 负责收集集群性能的统计数据，包括存储利用率、当前性能指标和系统负载。MDS 存储有关文件系统目录和文件元数据的信息。

Ceph 使用了 CRUSH(Controlled Replication Under Scalable Hashing）算法。CRUSH 算法是将 PG 映射为 OSD 的过程，一个文件被写入到 Ceph 时，会经过如下流程：

（1）文件会先被映射为对象，获得一个全局唯一的 object id（OID）；

（2）对 OID 进行简单哈希，得到一个 PG（Placement Group）编号，即 PG_ID。PG 是抽象的存储节点，不随着物理节点的加入或则离开而增加或减少，因此数据到 PG 的映射是稳定的，这是逻辑层的概念；

（3）将 PG_ID 作为 CRUSH_HASH 的输入，通过 CRUSH_HASH(PG_ID, OSD_ID, r) 得出一个随机数。对于所有的 OSD 用他们的权重乘以每个 OSD_ID 对应的随机数得到乘积，选出乘积最大的 OSD，PG 就会保存到这个 OSD 上；

（4）上述的计算过程都是在 Ceph 客户端完成的，之后 Ceph 客户端将会直接定位具体的 OSD，进行数据的读写。

CRUSH 算法为每个设备分配一个权重值，其目标是逼近 I/O 请求的均匀概率分布。而这个权重值并没有一个普遍适应的值，是根据最佳实践来调整的，一个原则就是这个权重值要能方便 Ceph 将更多的数据分配给较大的驱动器，而将更少的数据分配给较小的驱动器。

Ceph 在接口层提供了文件、对象、块存储等能力，但是在存储时，都是作为对象存储。Ceph 提供了 RESTful API 接口，允许用户直接在对象级别存储数据，适用于大规模数据存储，如云平台的存储后端。Ceph 可以提供块设备接口，允许虚拟机或裸机服务器直接连接并使用 Ceph 存储集群。Ceph 的分布式文件系统（CephFS）将文件和目录作为对象存储分布式对象存储设备上。CephFS 利

用 MDS 优化了文件操作的性能，可以作为标准的 POSIX 兼容文件系统被挂载使用。

2.2.3　Lustre

Lustre 是一个高性能的分布式文件系统，被广泛应用于大规模并行计算环境，能够连接和管理通过高速网络互联的大量计算节点上的数据，常见于超级计算机和大规模数据中心。

Lustre 文件系统的架构的主要组件包括：元数据服务器和元数据目标、对象存储服务器和对象存储目标、以及客户端。元数据服务器负责存储和管理文件的元数据，如文件名、目录结构和权限信息等。每个元数据服务器上会有一个或多个元数据目标，元数据目标实际存储元数据。对象存储服务器是负责存储文件数据本身的服务器，每个对象存储服务器可以管理多个对象存储目标，对象存储目标是实际保存文件数据的存储设备，文件切分为多个对象，这些对象分布存储在多个对象存储目标上。客户端通过高速网络与元数据服务器和对象存储服务器通信，实现对文件系统的访问和操作。

在 Lustre 中，目录对象和文件对象的布局由元数据服务器决定。文件对象和目录对象的分配采用循环算法或加权算法。对于顶级目录，当空闲的 inode 和块的数量平衡时（默认情况下，不超过 5% 差异），使用循环算法选择下一个元数据目标进行目录创建或下一个对象存储目标进行对象创建。如果目录位于根目录以下三个层级，或者元数据目标之间不平衡，使用加权算法在空闲 inode 和块更多的元数据目标中随机选择。对于文件条带的分配，当对象存储目标之间的空闲空间差异小于 17% 时，采用循环算法选择下一个对象存储目标写入条带。元数据服务器会周期性调整条带化布局，以消除某些异常情况，避免应用程序过度偏好特定对象存储目标。当对象存储目标之间的空闲空间差异超过 17% 时，采用加权随机分配，优先选择空闲空间更多的对象存储目标，但可能会降低 I/O 性能，直到空间再次平衡。这两种算法和限制确保了文件和目录对象在 Lustre 文件系统中的均衡分配和有效管理，以提供高性能的文件访问和存储服务。

2.2.4　BeeGFS

BeeGFS 是一个现代的、高性能的并行文件系统，专为大规模数据存储和高速数据访问设计。它被广泛应用于科学研究、高性能计算和企业级数据中心等领域。BeeGFS 以其出色的扩展性、卓越的速度和简单的安装与管理而著称，能够轻松与各种硬件配置集成，从而提供无缝的并行数据处理能力。

BeeGFS 架构图如图 2-1 所示，包括多个组件：管理服务器、元数据服务器、存储服务器和客户端。管理服务器负责整个文件系统的管理和监控。元数据服务器处理有关文件名称、目录结构、权限和文件布局的元数据。存储服务器存放文件数据，可以很容易地添加更多的存储服务器来扩展文件系统的容量。客户端则是安装在用户工作节点上的组件，它允许用户通过标准的 POSIX 接口访问文件

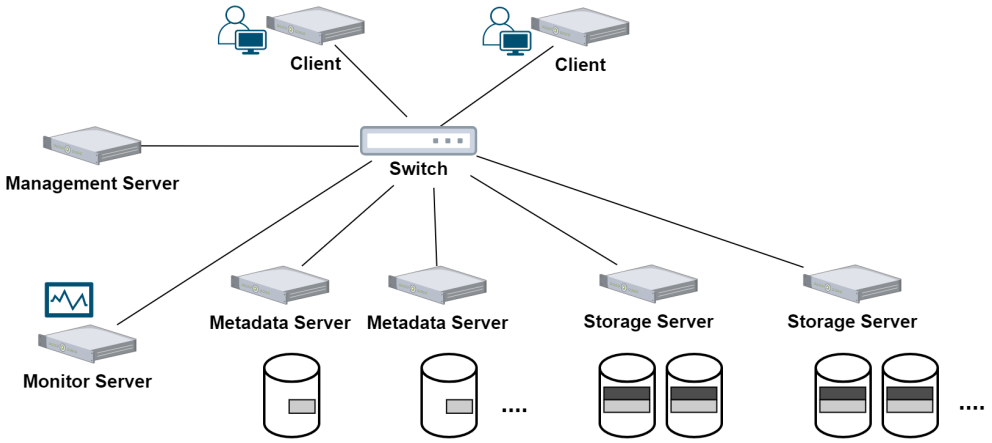


图 2-1　BeeGFS 文件系统基本架构

Figure 2-1　The system architecture of BeeGFS

系统。这种分离的模块化设计使得 BeeGFS 能够在不同的服务器上并行处理元数据和数据请求，从而实现高性能和可伸缩性。

BeeGFS 采用了动态条带化技术，可以灵活地调整每个文件的条带宽度和条带计数，从而优化不同类型的 I/O 操作。在分配文件到存储设备时，BeeGFS 利用其内部算法均匀地分配数据，以避免热点并确保高效利用存储资源。此外，BeeGFS 的算法还包括故障检测和自我修复机制，确保文件系统的稳定性和数据的完整性。

BeeGFS 支持多种冗余机制，包括镜像和纠删码（Erasure Coding）。镜像可以为目录、文件或者整个文件系统创建副本，防止数据丢失。BeeGFS 的冗余机制是在文件系统层面实现的，这意味着它与底层硬件和 RAID 配置无关。这使得 BeeGFS 可以在各种不同的硬件和存储配置上工作，并提供一致的冗余保护。

总体而言，BeeGFS 是一个强大的并行文件系统，具有灵活、高效通用等优点，能够来满足高性能计算和大数据处理的要求，以轻松地扩展以适应不断增长的数据量，同时提供高速访问和强大的数据保护机制，使其成为科研和工业界的理想选择，也是本文选择的分布式文件系统。

2.2.5　GlusterFS、Ceph、Lustre、BeeGFS 对比

随着 ARM 架构在服务器领域的渗透率不断提高，越来越多的厂商和组织开始关注在 ARM 平台上开发和部署分布式存储系统。上述开源分布式文件存储项目也都进行了探索，Ceph、GlusterFS(Gudu 等, 2016)、Lustre(Pedretti 等, 2020) 和 BeeGFS(BeeGFS, 2022) 已经开始支持 ARM 架构，并取得了一定的成果。

Lustre 适用于高性能计算领域，但由于其采用了串行元数据访问模型，导致元数据扩展能力有限，并发文件操作的效率较低。此外，它仅能通过 RAID 提供冗余功能。Ceph 以对象存储为核心，同时提供块存储、文件存储、对象存储功能，但其文件系统功能相对较弱，且复杂的软件栈使得性能相对较弱且不稳定，也使得部署及运维较为复杂。GlusterFS 没有中央的元数据管理节点，因此不存

在单点故障和性能瓶颈的问题。然而，由于元数据分布在每个块上，导致了一致性处理方面的困难。此外，GlusterFS 在处理小文件写入和递归操作时性能较差，难以应用于大型文件系统的交互操作。

相比之下，BeeGFS 具有轻量级的代码和快速的元数据处理能力，适合部署在 21CMA 的数据存储系统——基于 ARM 架构的低功耗分布式存储系统及终端设备上。BeeGFS 能够分离元数据并提供快速的元数据访问，同时简化了元数据操作的分配。它可以按需添加元数据节点，具有良好的系统扩展性，并且设置比 Lustre 的元数据服务器更简单。因此，本文选择 BeeGFS 作为分布式存储运维管理系统的核心组件，为基于 ARM 架构的低功耗分布式存储系统及终端设备提供高效可靠的存储解决方案。

2.3　小结

本章分析了选择 21CMA 数据存储设备文件系统时需要考虑的因素，对现有的存储技术从网络连接方式、数据访问方式、架构方面进行了讨论，得出了选用分布式文件系统的结论。在此基础上，本章详细介绍了几种流行的分布式文件系统：GlusterFS、Ceph、Lustre 和 BeeGFS，并对它们进行了比较分析，阐述了为何使用 BeeGFS 作为 21CMA 数据存储的分布式文件系统。

第 3 章　21CMA 数据存储集群运维管理方法研究

21CMA 数据存储集群的核心设备是低功耗分布式存储系统及终端设备。低功耗分布式存储系统及终端设备将 BeeGFS 部署在 ARM 架构的树莓派上，具有高存储速率、高存储容量、低成本的优点，但还不能满足 21CMA 数据存储集群对部署高效性和一致性、运行稳定性、管理便捷性的需求。本章将分析 21CMA 的数据量和 21CMA 的地理位置、运维条件，探究 BeeGFS 分布式文件系统和低功耗分布式存储系统及终端设备需要改进之处，最后设计出低功耗分布式存储系统及终端设备运维管理系统的架构和功能模块。

3.1　21CMA 数据存储集群需求分析

升级后，21CMA 的每个子阵每小时预计产生 TB 级的数据，81 个子阵每小时将产生数 TB 的数据。这就要求存储设备有 GB/s 的数据存储速率和 TB 级别的数据存储容量。因此，整个 21CMA 存储集群需要多个低功耗分布式存储系统及终端设备。而每个低功耗分布式存储系统及终端设备有多个服务节点和硬盘，21CMA 数据存储集群将是一个有上百个节点和硬盘的庞大集群。

低功耗分布式存储系统及终端设备从初始化、启动到后期运行，涉及一系列操作。首先，需要将 BeeGFS 的各个服务安装和部署到集群中的各个节点上。随后，进行存储池和伙伴组的配置工作。最后，依次启动管理服务、元数据服务、存储服务以激活 BeeGFS。在 BeeGFS 服务启动并投入使用之后，可能会出现需要扩展系统以容纳新节点、应用软件更新等文件系统操作。整套系统最多可达 81 个集群，每个集群有多个服务，比如元数据服务、存储服务等，以及一些附加的存储系统优化方法。在多个集群和多个节点上安装和配置 BeeGFS 时，需确保每个节点上的 BeeGFS 服务都遵循相同的配置参数和部署流程，即需要保证 BeeGFS 部署的一致和可复现性。此外，需要尽量降低部署所需的时间和人力成本。

根据 Google 数据中心一年内服务器的故障统计显示，数据集群中出现频率最高的故障依次是：硬盘故障、单机故障，DNS 故障和路由器故障，以及其他极少的个别情况 (杨传辉, 2013)。DNS 故障通常持续 30 秒可自动恢复，路由器故障往往要通过重启解决。因此，要保证一个数据中心的可靠性、稳定性，硬件故障和网络故障尤其需要关注。每一套低功耗分布式存储系统及终端设备都是由多组树莓派和硬盘组成的，整个 21CMA 数据存储集群将有数百个树莓派和硬盘，硬件故障无法避免。当前使用的 BeeGFS 文件系统虽然在性能上表现优异，但其故障处理的能力有限，需要进一步扩展其故障处理能力。

分布式存储集群监控是指对分布式存储系统中的各个节点和组件进行实时监测和管理的过程。它旨在监控存储集群的稳定性、性能等，并及时发现和解决

潜在的问题，以保障数据的安全和高效访问。分布式存储集群监控通常包括系统状态监控、性能监控、容量监控、故障监控。对于 21CMA 数据存储集群而言，要能随时查看集群的节点数量、每个节点的磁盘配置、集群整体的 IO 性能和存储容量等。在运维管理期间，还需要获得集群运行情况的即时反馈，以及把握系统整体运行态势。对集群节点、磁盘数量的监控属于系统状态监控，对集群 IO 速率的监控属于性能监控，对集群的存储容量使用情况的监控属于容量监控，对软硬件故障的监控和对故障恢复日志的记录属于故障监控。而 BeeGFS 原有的监控功能虽然包括系统监控和性能监控，即节点的状况和吞吐量，但是缺乏对集群总吞吐量、容量以及集群故障日志的监控。此外，还需要一个集中的监控服务来展示整个 21CMA 存储集群的运行状况。因此，BeeGFS 的监控功能需要进一步扩展。

根据上述分析，低功耗分布式存储系统及终端设备的软件能力还需要在部署流程、故障恢复能力、集群监控模式三个方面进行进一步提升。

3.2 21CMA 存储设备运维管理系统功能需求

根据以上对 21CMA 数据存储设备需求的分析，系统应当能实现以下基本功能：

1. **交互功能**：系统应具备用户友好的界面，提供直观的操作流程。用户通过该界面管理集群和 BeeGFS 文件系统，执行如启动、停止、配置 BeeGFS 文件系统及集群的操作。

2. **自动化部署功能**：系统应具备自动化部署能力，能够依据用户定义的节点角色自动执行 BeeGFS 的安装和配置过程。

3. **软硬件管理功能**：系统应具备灵活的软硬件管理系统。系统应支持用户通过前端界面请求添加或移除节点、调整 BeeGFS 伙伴组和存储池配置，后端应能处理这些请求并返回操作

4. **自动化故障恢复功能**：在节点发生故障时，系统应能快速诊断和分类故障类型的能力，并自动采取相应的恢复措施。对于软件故障或网络故障，需重启主机或服务；如果是某个硬盘或者服务节点出现硬件故障，需要及时替换该硬件，以最小化存储系统的故障时间。

5. **集群监控及监控数据可视化**：系统应能实时监控每个集群的运行状况，包括节点状况、系统性能、存储容量等。系统应提供实时的监控视图，以便当系统出现问题时，用户需要能够快速收到警告以便于及时处理。例如，根据存储容量使用情况调整存储策略或调整存储设备。此外，系统应提供多集群统一监控视图，以便于同时监控 21CMA 的多个存储集群。

综上，满足 21CMA 存储设备运维管理需求的关键在于构建一套具备自动化部署、自动化故障恢复能力以及便于软硬件管理及集群监控的运维管理系统。

3.3　BeeGFS 应用及功能分析

本节将介绍 BeeGFS 部署流程、现有命令行工具、故障恢复机制、监控服务，总结现有技术基础并指出需要进一步改进之处。

3.3.1　BeeGFS 部署流程

在 Linux 集群上手动安装和部署 BeeGFS 文件系统涉及一系列细致的步骤。首先，需要准备集群中的各个节点，并确保其兼容 BeeGFS 系统。其次，配置每个节点的主机名和 IP 地址映射关系，并将此信息同步到所有集群节点。在每个节点上，下载并安装相应的 BeeGFS 软件，并创建必要的数据存放目录。

部署管理服务。运行命令“‘/opt/beegfs/sbin/beegfs-setup-mgmtd -p mgmtd_path‘”，其中 ‘mgmtd_path‘指定了管理服务的数据存放目录。

部署元数据服务，运行命令 “‘/opt/beegfs/sbin/beegfs-setup-meta -p meta_path -s meta_id -m mgmtd_host‘” ，其中 ‘meta_path‘为元数据服务的数据存放目录，‘meta_id‘为元数据服务的唯一标识，‘mgmtd_host‘为管理服务节点的主机名或 IP 地址。

部署存储服务。运行命令“‘/opt/beegfs/sbin/beegfs-setup-storage -p storage_path -s storage_host_id -i storage_id -m mgmtd_host‘” ，其中 ‘storage_path‘为存储服务的数据存放目录，‘storage_host_id‘为存储服务的唯一标识（通常使用 IP 地址命名），‘storage_id‘为存储服务 ID，‘mgmtd_host‘为管理服务节点的主机名或 IP 地址。部署监控服务，修改 ‘/etc/beegfs/‘目录下的 ‘beegfs-mon.conf‘文件，设置 ‘sysMgmtdHost=mgmtd‘，其中 ‘mgmtd_host‘为管理服务节点的主机名或 IP 地址。

最后，部署客户端。运行命令“‘/opt/beegfs/sbin/beegfs-setup-client -m mgmtd_host‘”，其中 ‘mgmtd_host‘为管理服务节点的主机名或 IP 地址。完成以上配置后，依次启动管理服务、元数据服务、存储服务、客户端和监控服务。

从上述部署流程分析可知，BeeGFS 的部署过程较为复杂，涉及设置元数据服务和存储服务的唯一标识符（ID），以及配置各节点的管理参数，此外还需要修改监控服务器中的配置文件。因此，需要开发运维管理工具以配置降低部署成本和提高效率。

3.3.2　BeeGFS 文件条带化原理

BeeGFS 具有 BuddyMirror 和 Raid0 两种模式，两种不同的存储模式通过文件条带实现，如图 3-1(Heichler, 2014) 所示。该图展示了 BeeGFS 中文件的条带模式以及它与 RAID0 和 BuddyMirror 两种存储模式的关系。

在右半部分，“/beegfs/unmirrored” 路径下的文件（文件 #3）没有使用镜像。文件的数据条带直接分散在各个存储目标上，没有冗余复制，即 RAID0 存储模式，它通过条带化的方式分布数据以提高性能，但不提供冗余。

在左半部分，“/beegfs/mirrored” 路径下的文件：文件 #1 和文件 #2 则使用了镜像存储模式。在这种模式下，数据被复制到两个不同的存储目标组（BuddyGroup），

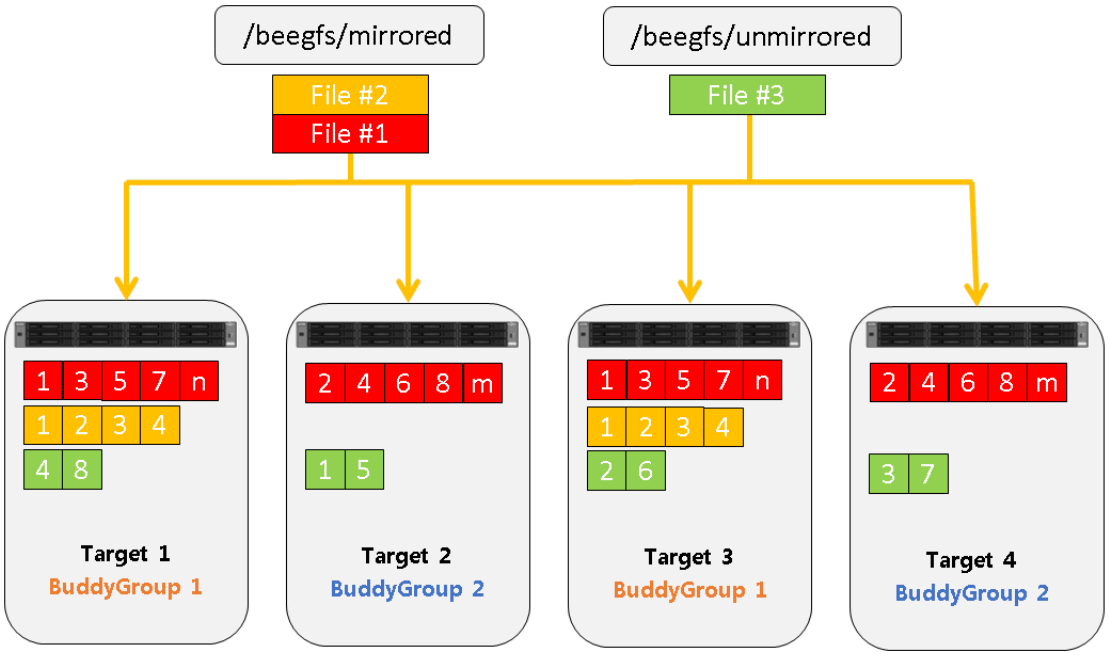


图 3-1　故障检查及恢复流程图

Figure 3-1　Flow chart of failure checking and recovery

以实现数据冗余。例如，文件 #1 的数据条带被分布在 BuddyGroup 1 的 Target 1 和 BuddyGroup 2 的 Target 2。这提供了数据的高可用性，任何一个存储目标发生故障，其镜像仍然可以提供数据读写。

对于 BeeGFS 的两种存储模式，RAID0 存储模式不能提供冗余，BuddyGroup 存储模式只提供了有限的冗余。BuddyGroup 存储模式虽然能保护免于单个存储节点的故障，但如果该伙伴组再次出现故障该数据块将无法恢复。因此需要在原有故障恢复的基础上进行二次开发。

3.3.3　BeeGFS 控制工具

BeeGFS 提供了 beegfs-ctl 控制工具，该工具是 beegfs-utils 软件包的一部分，用于更改和查询配置、管理节点和存储目标、操作文件和目录以及监控系统性能。以下是 beegfs-ctl 具体的命令行参数及其对应功能的详细说明：

首先是节点和存储目标管理方面，使用“‘–listnodes‘”和“‘–listtargets‘”命令可以列出已注册的元数据节点、存储节点或存储目标。此外，使用“‘–removenode‘”和“‘–removetarget‘”命令可以移除节点或存储目标。

在文件和目录管理方面，可以通过“‘–createfile‘”和“‘–createdir‘”命令创建文件和目录，使用“‘–migrate‘”命令将文件迁移到其他存储服务器。

对于性能监控和基准测试，可以使用“‘–serverstats‘”和“‘–clientstats‘”命令显示服务器和客户端的 I/O 统计信息，使用“‘–storagebench‘”命令在存储目标上执行基准测试，以评估存储性能。

在配额管理和镜像功能方面，使用“‘–getquota‘”和“‘–setquota‘”命令可以

显示和设置用户或组的配额限制，使用“‘–addmirrorgroup‘”和“‘–startresync‘”命令可以添加镜像伙伴组并启动存储目标或元数据节点的数据同步。

最后，关于存储池管理，可以使用“‘–listpools‘”命令列出所有的存储池，通过“‘–createpool‘”命令创建一个新的存储池，使用“‘–deletepool‘”命令删除现有的存储池，使用“‘–assignpool‘”命令将存储目标分配到特定的存储池。

在本研究中，利用 beegfs-ctl 工具对存储池进行管理以分配存储资源，并通过配置伙伴组实现存储的镜像模式。然而，beegfs-ctl 目前不支持删除已存在的伙伴组或从伙伴组中删除节点，beegfs-ctl 的- -migrate 子命令也无法在发生故障时从故障节点迁移数据。因此，为了在现有故障恢复功能基础上进行二次开发，本文将探索其他方法以调整故障伙伴组的主次节点配置并恢复数据。

3.3.4　BeeGFS 监控服务

BeeGFS 监控默认采用时序数据库 InfluxDB 存储元数据节点和存储节点的信息，默认生成的数据库名为 beegfs_mon，其数据表及结构详见图 3-2。

图 3-2　beegfs_mon 数据库结构

Figure 3-2　The database structure of beegfs_mon

(1) **表 meta：**记录元数据节点的运行状况和工作负载。其中，isResponding 和 nodeID 用于监控节点的响应状态和身份，indirectWorkListSize 和 directWorkListSize 提供节点工作列表的大小，反映其处理能力和负载。

(2) **表 highResMeta：**提供更高分辨率的元数据节点性能数据。其中，busyWorkers 和 queuedRequests 反映工作线程忙碌程度和请求队列长度，netSendBytes 和 netRecvBytes 测量网络流量，workRequests 统计工作请求。

(3) **表 metaClientOpsByNode：**按节点统计客户端操作，如确认操作（ack）和请求元数据（mdsInf）的次数，node 和 sum 字段用于识别节点并总结其请求活动。

(4) **表 metaClientOpsByUser：**按用户统计客户端操作，包括操作次数和类型，用于分析特定用户的行为和系统使用模式。

(5) **表 storage：**跟踪存储节点的基本运行状态和磁盘空间，用于监控存储容量和响应状态。

(6) **表 highResStorage：**提供详细的存储节点性能数据，包括磁盘读写字节、网络流量和工作请求等，用于详细分析节点性能。

(7) **表 storageClientOpsByNode：**按节点统计存储请求操作，用于分析各个节点的数据访问模式和负载情况。

(8) **表 storageClientOpsByUser：**按用户统计存储请求操作，用于分析用户级的数据访问和活动。

(9) **表 storageTargets：**跟踪特定存储目标的磁盘空间和 inode 的使用情况。其中的 targetConsistencyState 记录存储一致性状态，用于维护数据完整性和可靠性。

从上述数据表分析可以看出，尽管 BeeGFS 原有监控服务已经很丰富，但缺乏对故障恢复操作的记录，也无法一次性查看所有节点的运行状况及系统存储总量及使用情况。因此，需要一个定制化监控页面补充现有监控功能的局限，以实现对集群所有运行信息的全面实时监控，并对存储容量及故障恢复日志进行详细记录，以提升系统的管理效率。

3.4 系统架构设计

基于以上分析可知，部署在每个集群上的运维管理系统首先需要构建一个 Web 框架使得用户可以与系统进行操作交互，然后在此基础上集成服务于低功耗并行存储集群的模块。具体模块包括：部署及软硬件管理模块、自动化故障恢复模块、监控数据可视化模块。此外，系统应该开发一个统一监控网站，以实现多集群的管理及监控需求。通过这些模块的协同工作，得以构建一个能够有效管理和维护低功耗分布式存储系统及终端设备的平台，确保其稳定、高效地运行。

图 3-3 展示了 21CMA 低功耗分布式存储系统及终端设备的运维管理系统模块及其层次结构。系统的上层部署于每个集群，包括 Web 框架及其相关模块，这些模块涵盖了部署与软硬件管理、自动化故障恢复以及监控数据可视化三大功能。具体来说，部署及软硬件管理模块负责记录软硬件数据信息，自动化故障恢复模块负责自动化故障恢复和记录故障恢复日志，而监控数据可视化模块则汇总并展示监控数据。这些数据随后被存储于数据库中。在系统的下一层，统一监控服务模块负责存储来自单个集群数据库的数据，并在前端页面展示这些汇总数据。

此外，模块化设计有利于系统功能的扩展和维护。每个模块都有明确的职责

边界，可以方便地增加新模块或对现有模块进行升级和优化。如果需要增加新的功能，直接添加一个新的模块即可，而不会影响其他模块的运行。系统还采用松耦合设计。各个功能模块之间通过标准化的接口进行数据交换和服务调用，降低了模块间的依赖度，使得单个模块的替换和迭代更加方便。这样的设计进一步提高了系统的可扩展性。

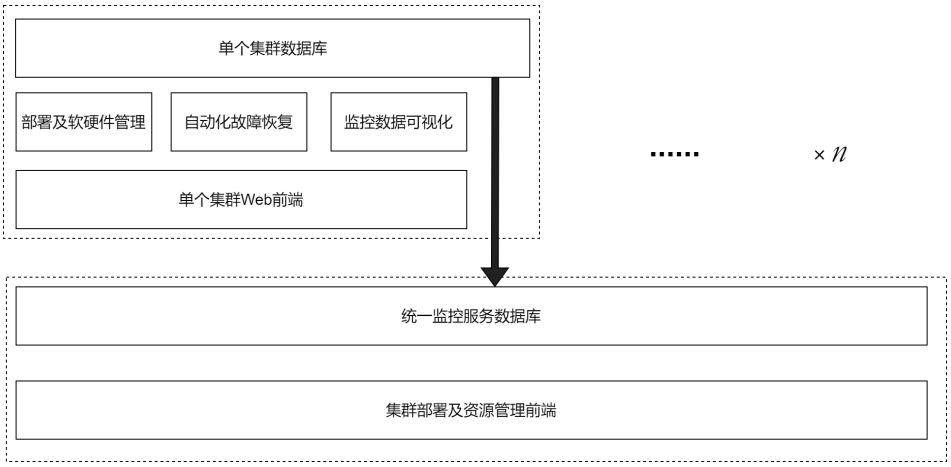


图 3-3　21CMA 存储运维管理系统架构图

Figure 3-3　Architecture diagram of 21CMA storage operation and management system

3.5　模块功能设计

以下将针对系统需求及模块划分详细描述系统的功能设计：

（1）部署模块

- 节点操作：为用户提供对单个节点操作的接口，包括节点的增加、删除、基本信息修改。
- 磁盘操作：为用户提供对单个磁盘操作的接口，包括磁盘增加删除、磁盘挂载及取消挂载、磁盘格式化等。
- SSH 密钥分发：生成 SSH 密钥，并将 SSH 密钥分发到各节点，确保各节点之间可以进行安全的通信，建立信任关系。
- 服务器资源分配：根据需求和系统性能要求，在集群中分配服务器资源，包括确定每个节点的存储容量和角色。
- 扫描磁盘：自动扫描集群中的磁盘设备，识别集群中的磁盘信息，包括 UUID、挂载路径、总容量以及使用容量等。
- 软件分发及安装：将所需的软件分发到各节点，并进行安装和配置，确保每个节点都能正确运行所需的存储软件。
- 其他操作：为用户提供设置伙伴组，以及存储池增加、删除、修改的接口。
- 此外，本模块需要兼顾全新集群的初始化、部署和后期人为更改软硬件设置。

（2）故障恢复模块

- 及时发现故障：通过实时监控系统状态和节点健康状况，能够快速检测到各种故障，如节点宕机、磁盘故障等。
- 故障类型反应：根据故障类型采取相应的恢复措施，例如自动将故障节点从集群中剔除，并启动备用节点来接管故障节点的工作。
- 记录故障恢复日志：对每次故障恢复过程进行记录，包括故障发生时间、恢复措施、恢复时间等信息，用于后续故障分析和优化。

（3）单个集群监控模块

- 存储使用情况：实时监测集群的存储使用情况，包括已使用空间、剩余空间等，以便对存储资源进行合理管理和规划。
- 读写速率：监控集群的读取和写入速率，以评估存储性能，并及时发现潜在的性能瓶颈或异常情况。
- 节点总数及运行节点：记录集群中节点的总数，并监测当前正常运行的节点数量，以确保集群的可用性和稳定性。
- 故障节点和备用节点数目：实时追踪故障节点的数量，并监控备用节点的可用性和数量，以确保系统的冗余和容错能力。
- 故障恢复情况：记录故障发生时的恢复过程和时间，以评估系统的故障处理效率和可靠性。

（4）多集群统一监控模块

- 综合信息展示：在统一监控网页上展示每个集群的存储使用情况、读写速率、节点运行状况、故障节点数量等信息，以提供全局的存储系统状态概览。

3.6 模块部署方案

由于监控节点负载较低，并且可以直接获取监控数据无需再次分发，所以本文最终选择将单个集群的运维管理系统部署在监控节点上。统一监控服务独立于其他模块，可以部署在任何地方。最终 21CMA 数据存储系统的运维管理系统如图 3-4 所示。由上至下分别为 21CMA 数据采集模块、存储设备和统一监控系统。21CMA 子阵采集的数据经过数字后端处理后，分别写入到独立的存储设备中。每个存储设备包括管理节点、元数据节点、存储节点及监控节点，并以树莓派作为服务节点。本系统以 BeeGFS 开源版本为基础，增加一种新的节点类别即备用节点，用以实现系统高可用性。本系统的标准化部署及故障恢复模块安装在存储设备的监控节点。统一监控模块部署在远程服务器上，对监控节点数据进行统一展示。

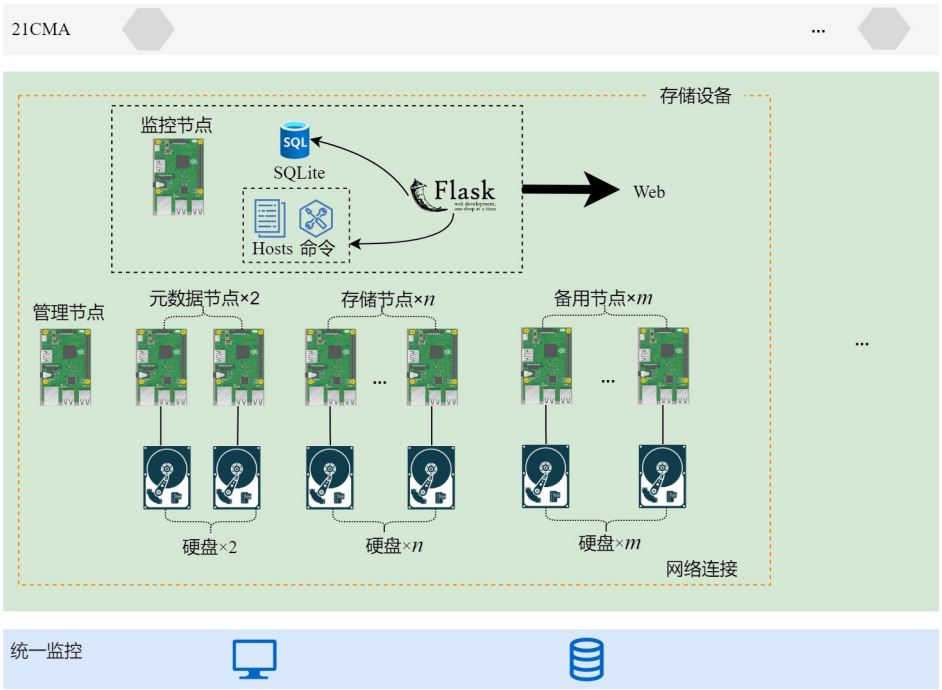


图 3-4　21CMA 存储系统架构图

Figure 3-4　Storage system architecture for 21CMA

3.7　小结

本章对 21CMA 数据存储集群的数据量和运维条件进行了分析，详细阐述了系统的需求来源与功能需求。基于功能需求，本文设计了系统架构以及部署与软硬件管理模块、自动化故障恢复模块、单个集群监控模块、统一监控服务模块四个功能模块。在接下来的章节中，将依据已确定的架构和模块设计，详细展开系统数据库及各个模块的设计与实施过程。

第 4 章　21CMA 数据存储设备运维管理系统实现

本章以第三章 21CMA 数据存储集群需求分析和运维管理系统的概要设计为基础，深入展开系统的详细设计与实现，包括数据库的详细设计与实现和四个模块的设计与实现。

4.1　系统实现技术需求及选择

本节将介绍在项目实现中所采用的工具，并描述工具在系统中的功能以及各工具之间的依赖关系。

4.1.1　网站实现技术需求及选择

本系统需要持续监控和记录低功耗分布式存储系统及终端设备的运行状况，并维护其运行。因此，系统需要集成部署模块、故障恢复模块、监控组件等，并且在后续可能会持续集成或部署新的组件。Flask 是一个轻量级的 Web 框架，易于扩展，可以集成数据库、认证、日志记录等功能，选择 Flask 既能满足当前需求，也便于将来继续扩展系统功能。

单个集群运维管理系统数据库的流量来自于初期配置集群、节点、磁盘信息和后期运行时的系统性能、系统故障监控，流量较低。因此可以选择轻量的数据库。SQLite 是一个流行的嵌入式 SQL 数据库引擎，具有轻量、零配置、数据类型灵活的优点，运维管理系统主要的流量选择 SQLite 既能满足需求，也能尽量降低软硬件成本。

统一监控服务收集单个集群的监控数据需要通过网络来实现。SQLite 是一个嵌入式数据库，缺少用于网络中的数据传输的内置支持。相比之下，PostgreSQL 作为一个功能强大的开源对象关系型数据库系统，不仅支持网络数据传输，而且提供了诸如高级索引、并行查询处理和高效数据缓存等优化功能，这些特性使得它特别适合处理大规模数据集。此外，PostgreSQL 的优秀扩展性使其能够有效管理随时间不断增长的数据量。它的分区表和表继承功能允许数据库模式随时间和业务需求变化而灵活调整，从而满足未来可能增加的更多集群监控需求。因此，对于需要通过网络传输并保存大量数据的场景，PosgreSQL 是合适的选择。

集群与统一监控服务器连接需要一个适配器。Psycopg2 是一个开源的 Python 数据库适配器，能够连接和操作 PostgreSQL 数据库，是 PostgreSQL 的官方推荐适配器之一。Psycopg2 提供了一个高效、稳定的接口，使开发人员能够与 PostgreSQL 数据库进行交互。Psycopg2 通过提供 Python 对 PostgreSQL 数据库的访问功能，使得开发人员能够使用 Python 语言进行数据库操作，如执行 SQL 查询、插入、更新和删除数据等。本文选择 Psycopg2 作为适配器，与远程数据库连接，实现集群数据向统一监控服务器的传输。

4.1.2 远程配置管理实现技术需求及选择

本系统需要能够对几十个甚至上百个节点进行批量操作，并保证部署的一致性，需要借助配置管理工具来改善部署流程和管理方式。在 21CMA 数据接收与存储场景中，低功耗分布式存储系统及终端设备采用树莓派作为服务节点。树莓派内存容量较小，在选择部署工具时，需要尽量选择附加成本低的工具。Ansible 是一种自动化工具，用于大规模多节点的操作和管理。Ansible 为无代理架构，不需要在节点上安装额外的软件，从而降低了系统的开销和成本，适合在资源有限的设备上使用。Ansible 的批量操作功能可以实现对数十个甚至数百个节点的行统一管理，确保了部署的一致性。通过 Ansible 自带的 command 工具，用户可以指定在所有节点上执行的任务，Ansible 将负责在所有指定的节点上一致地应用这些配置。此外，Ansible 支持动态清单，通过修改 inventory 可以动态添加或者删除节点，确保了清单的实时性和准确性。因此，Ansible 可以满足 21CMA 数据存储集群的管理需求，能够在不增加过多负担的前提下，提供一个安全、可靠且易于管理的自动化环境。

在实际的部署过程中，监控节点需要修改远程节点的配置文件。Ansible 修改远程节点的能力有限，需要其他工具辅助。Ansible 调用 Paramiko 和 subprocess，实现多节点批量操作。

Python Paramiko 是一个用于在 Python 编程语言中进行 SSH 连接和文件传输的模块。通过 Paramiko 可以轻松地与远程服务器进行通信，并执行各种操作，如执行远程命令、上传、下载和修改文件等。Ansible 虽然便于在被控节点上调用各种命令，但是不便于修改被控节点的配置文件。而 Paramiko 提供了一个方便的接口。监控节点使用 Paramiko 的 paramiko.SSHClient() 类创建客户端，与管理服务所在节点建立连接，从而能够远程修改管理服务的配置文件。

在 Python 中，子过程调用通常是指使用 subprocess 模块来运行外部命令和程序。subprocess 模块可以用来创建新的进程，连接到它们的输入、输出、错误管道，并获取它们的返回码。本文主要使用的是 subprocess 模块的 Popen 类，这是一个较为底层的接口，可以用来执行复杂的进程管理任务如管道连接和输出流重定向。本文在远程和本地均通过 Popen 类来实现命令行调用，并通过其管道获取运行结果。

4.1.3 监控服务数据抽取技术选择

BeeGFS 监控服务模块默认使用 InfluxDB 数据库。InfluxDB 是一个开源时序数据库，适合用于处理和存储大量时序数据，如监控指标、应用程序遥测数据和物联网传感器数据。BeeGFS 采用 InfluxDB 做默认的时序数据库。为了获取数据，监控模块需要和 InfluxDB 数据库进行连接。influxdb-client 是一个 Python 库，非常易于安装，通过接口 client=InfluxDBClient(url,token,org) 即可连接，本文采用 influxdb-client 与与 InfluxDB 数据库交互。

4.2　数据库设计与实现

数据库设计的目标是确保系统的高效运行和灵活的数据管理，使得存储系统用户和管理员能够方便地进行系统部署、监控和维护，同时能够快速响应运行中可能出现的各种情况。本文的数据库包括单个集群运维管理系统数据库和统一监控网站数据库。

4.2.1　单个集群运维管理系统数据库设计与实现

单个集群运维管理系统数据库关注存储系统、集群、主机及其角色、存储池和目标等实体的详细追踪和管理。每个实体都拥有独特的属性如 IP 地址、设备名称、文件系统类型等。实体之间通过关系相连接，如主机与磁盘通过主机 ID 关联，主机可以包含多个磁盘。单个集群运维管理系统数据库具体实体、属性及关系如图 4-1 所示。具体细节如下：

(1) 实体

- Host（主机）实体代表存储系统中的服务节点，每个主机有自己的唯一标识符、IP 地址、用户名称、密码和 SSH 密钥状态。
- Disk（磁盘）实体代表主机内的磁盘，每个磁盘有自己的唯一标识符、UUID、设备名称、挂载路径、文件系统类型、磁盘容量大小和磁盘挂载状态。
- HostRole（主机角色）实体将主机与它们在分布式文件系统中担任的角色连接起来，每个主机角色都有自己的标识符，并记录了主机角色在存储系统担任角色的名称。
- StorageSystem（存储系统）实体代表集群安装部署的存储系统类型，包括唯一标识符和存储系统名称。
- StorageSystemRole（存储系统角色）代表存储系统中不同的服务节点，包括唯一标识符、所属文件系统 ID 以及名称。
- StoragePool（存储池）实体代表存储资源的集合，每个存储池都有自己的唯一标识符，记录了分布式文件系统中的 ID 和名称。
- StorageTarget（存储目标）实体代表存储池中的具体存储资源，每个存储目标都有自己唯一的标识符，记录了所属的存储池、在文件系统中的 num 以及状态。
- BuddyGroup（伙伴组）实体是存储池的一个部分，用于组织存储目标，每个伙伴组实体都有自己的唯一标识符，以及在存储池中的 ID、节点类型 ID、所属存储池 ID、主存储目标 ID、副存储目标 ID 以及是否镜像的标识。
- RecoveryLog（恢复日志）实体记录了主机的恢复操作，包括时间戳、操作类型和操作是否成功的标识。
- System（系统）实体记录了一个集群，每个集群有自己唯一的标识符，包括集群名称、集群存储系统类型。

(2) 实体之间的关系

- Host 与 HostRole 通过 host_id 相关联，表示一个主机可以同时被设置为多

种角色。

- Host 与 Disk 通过 host_id 相关联，表示一个主机可以挂载多个磁盘。
- HostRole 与 StorageSystemRole 通过 storage_system_role_id 相关联，表示每个主机角色都有对应的存储系统角色
- StorageSystem 与 StorageSystemRole 通过 storage_system_id 相关联，表示一种存储系统有多个不同类型的节点。
- StoragePool 与 StorageTarget 过 storage_pool_id 相关联，一个存储池可以包含多个存储目标。
- StoragePool 与 BuddyGroup 通过 storage_pool_id 相关联，一个存储中可以有多个伙伴组。
- StorageTarget 与 BuddyGroup 通过 buddy_group_id 相关联，一个伙伴组中有两个存储目标。
- RecoveryLog 与 Host 通过 host_id 相关联，记录了集群中节点的故障、对故障处理的日志。
- StorageSystem 与 System 通过 storage_system_id 相关联，表示集群的存储系统类型。

4.2.2 统一监控网站数据库设计与实现

统一监控网站数据库记录和管理每个集群的运行状况和 I/O 性能数据。核心实体包括集群和读写速率，其中集群实体记录如监控服务器 IP、状态更新时间戳等信息，而读写速率实体则记录每个集群的读写速率和相关时间戳。这些数据对于实时监控和历史性能分析非常关键。统一监控网站数据库实体如图 4-2 所示。具体细节如下：

- Cluster（集群）实体与存储系统关联，每个集群都有自己的唯一标识符、集群名称、监控服务器所在主机的 IP、记录时间戳和更新时间戳，以及描述集群状态的字段，包括总主机数、运行中的主机数、故障主机数和备用主机数。
- IO（读写速率）实体记录了存储节点的读写速率，每个集群都有自己的唯一标识符、读写记录时间、磁盘读速率、磁盘写速率、集群名称。

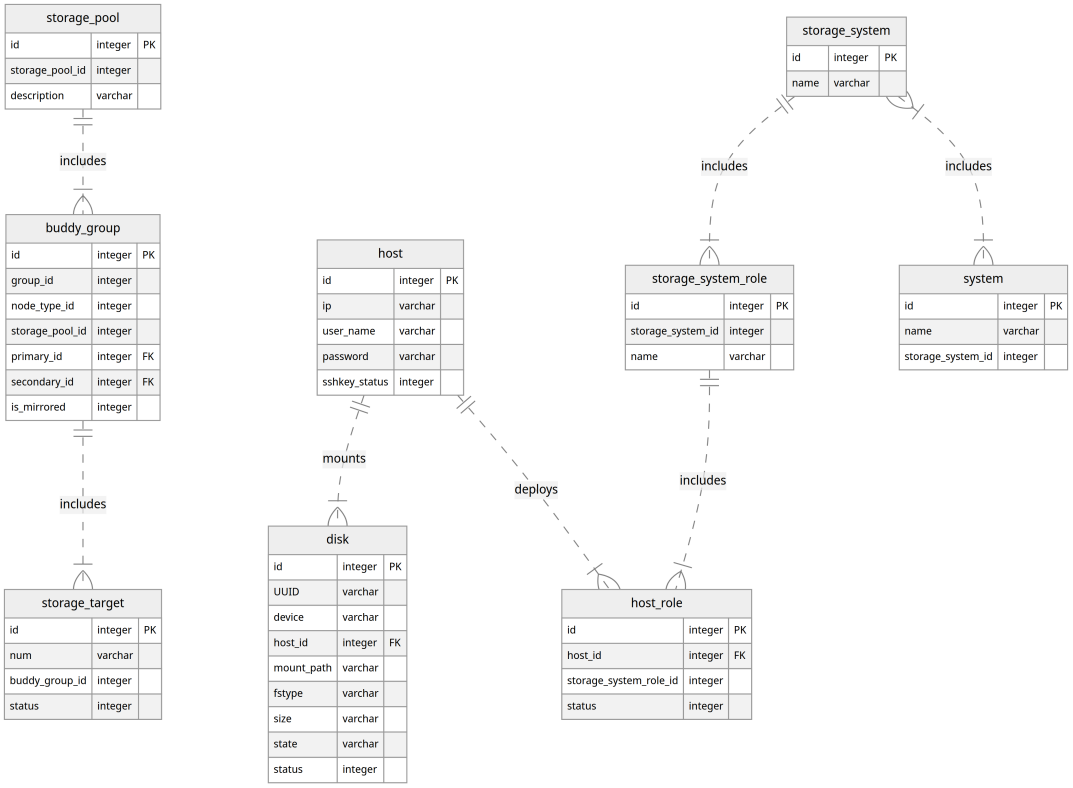


图 4-1　集群数据库 ER 图

Figure 4-1　ER diagram of cluster database

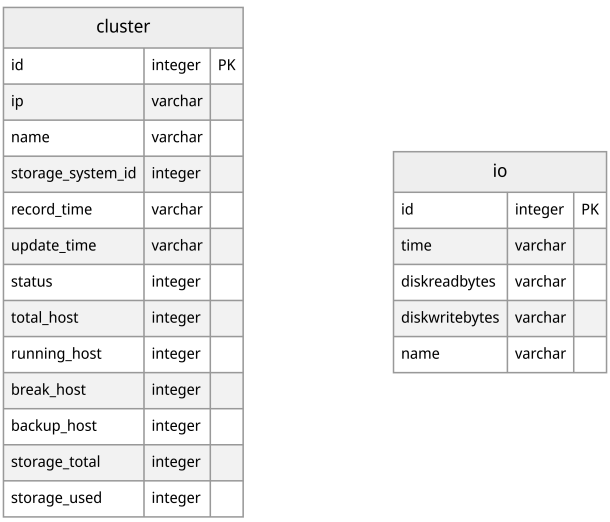


图 4-2　统一监控数据库 ER 图

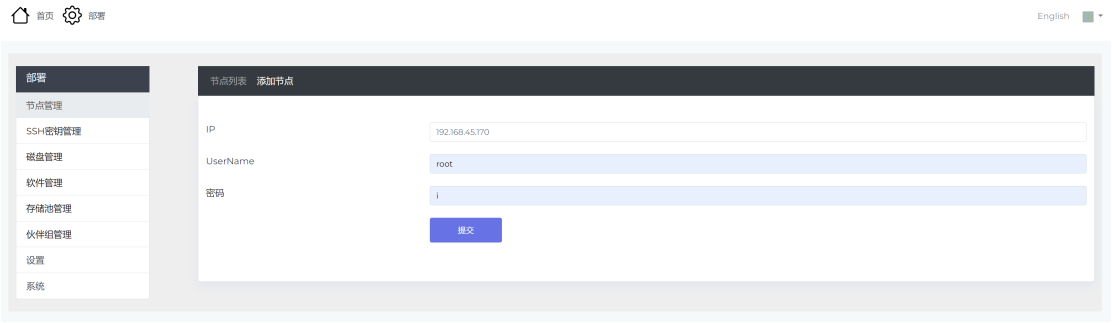
Figure 4-2　The ER diagram of the database is monitored in a unified manner

4.3 集群部署及软硬件管理模块实现

在单个存储设备内，存储节点、元数据节点及备用节点相互间通过网络连接。存储节点和元数据节点在所挂载的磁盘上存储数据。如何快速将存储设备组成一个逻辑一致的存储系统并作为一个整体运行是本模块的主要研究内容。集群部署及硬件管理模块的功能是快速组织多个节点，本节将详细描述其具体实现。

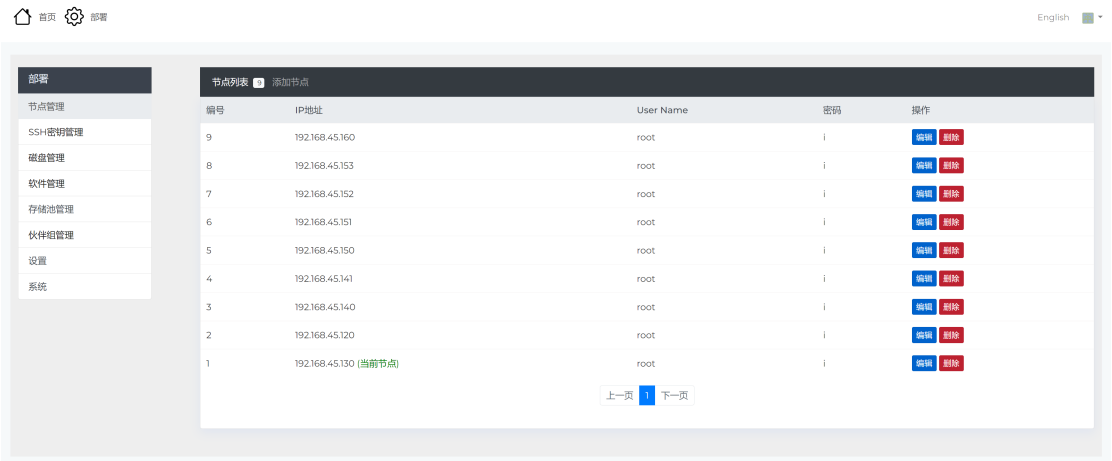
4.3.1 节点管理

本小节是对节点操作功能的详细设计与实现。模块提供了一个 Web 界面以及对单个节点进行操作的功能，包括添加、删除节点，以及修改节点信息，并提供了网页按钮和表单，用于用户提交节点添加删除或编辑请求，如图 4-3(a) 和图 4-3(b) 所示。用户在列表页点击按钮发起请求后，后端根据路由调用不同的视图。



(a) 节点添加页面

(a) Add page for host



(b) 节点列表页面

(b) List of host

图 4-3 节点展示及管理页面

Figure 4-3 Page of host display and management

第一个视图函数 host() 响应路由 “/host” 的 GET 和 POST 请求。在 GET 请求中，它主要用于展示主机列表或者添加/编辑主机的表单。如果是 GET 请求并

且操作类型为“list”，则会显示主机的分页列表，其中包括主机的详细信息。如果不是列表操作，它将提供一个表单以便用户可以添加或编辑主机。在 POST 请求中，它处理主机的添加或编辑操作。处理器首先会检查操作类型，如果是添加操作，它会收集表单数据并创建一个新的主机记录。如果是编辑操作，它将更新现有的主机。无论是添加还是编辑操作，完成后都会重定向用户回到主机列表页面。

第二个视图函数 delete_host() 响应路由“/delete_host”的 GET 请求，用于删除一个指定的主机。它通过查询字符串获取主机 ID，然后在数据库中找到并删除相应的 Host 记录以及与该主机相关联的 HostRole 和 Disk 记录。删除操作完成后，用户会被重定向回主机列表页面。

4.3.2　密钥生成及分发

SSH 密钥成功生成与分发能够建立节点之间的信任关系，对于后期的节点控制非常重要。本小节是对该功能的详细设计与实现。本功能提供了一个 Web 界面以及密钥生成及分发接口。如图 4-4 所示。用户在列表页点击按钮发起请求后，后端根据路由调用不同的函数。

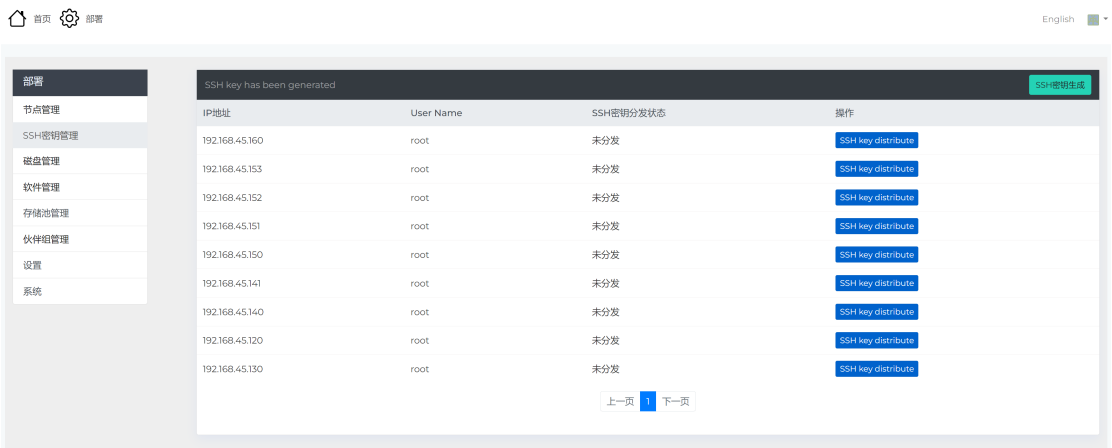


图 4-4　密钥生成及分发页面

Figure 4-4　Page of ssh-key distribute

1. 密钥生成视图函数 ssh_keygen() 响应路由“/ssh_keygen”，该函数的功能是自动化生成 SSH 密钥对。这个请求将会触发如下操作：收到请求后，该函数会调用 expect.spawn 启动一个子进程，执行“ssh-keygen -t rsa”命令来生成一个 RSA 类型的 SSH 密钥对。之后，函数使用 child.expect 捕获命令的输出，然后使用 child.sendline 发送输入到命令。本模块中按默认路径生成密钥。如果提示有同名密钥文件，则发送“y”来确认覆盖旧文件。最后函数将更新数据库中密钥信息并重定向回 SSH 密钥列表的页面。

2. 密钥分发请求处理：用户在网页上提交密钥分发请求。这个请求将触发后端执行以下操作：

(1) 将公钥分发至指定的节点：后端通过 SSH 协议与节点建立连接，并将生成的公钥复制到节点的相应位置，以便主控节点通过 Paramiko 库来实现与节点的通信。

(2) 检查密钥分发状态：后端检查每个节点的密钥分发状态。如果节点成功接收并保存了公钥，则将状态标记为“分发成功”；如果在分发过程中发生错误，则将状态标记为“分发失败”；如果节点已经存在密钥，并且需要覆盖，则将状态标记为“密钥已存在并覆盖”。

(3) 更新数据库：后端将节点的密钥分发状态更新到数据库中。

(4) 后端返回密钥分发结果给前端，以便在网页上显示。

4.3.3 磁盘扫描及管理

本小节是对磁盘扫描及磁盘操作功能的详细设计与实现。本功能提供了一个 Web 界面以及对磁盘进行操作的功能接口，如图 4-5(a) 及 4-5(b) 所示。用户在列表页点击按钮发起请求后，后端根据路由调用不同的函数。用户在前端发起磁盘扫描请求后，后端开始遍历元数据服务节点和存储服务节点，对列表中的每个角色执行以下操作：

(1) 使用 Ansible 和 lsblk 工具来获取关于磁盘的信息，列出块设备的信息，包括状态、大小和类型，并以键值对格式输出。

(2) 读取 Ansible 返回的结果，并开始逐行分析。通过循环遍历 Ansible 命令的输出，获取其挂载主机 IP、磁盘 UUID、磁盘挂载目录、磁盘总容量、磁盘已用容量、磁盘格式类型等信息。

(3) 根据节点列表通过 Ansible 调用远程命令接口，读取远程磁盘挂载信息，并根据磁盘挂载目录名称判断是否为外界磁盘。

(4) 检查磁盘是否已经记录在数据库中，如果不存在，则调用 Disk.add_disk() 函数添加新的磁盘记录；如果存在，则调用 Disk.edit_disk() 函数更新磁盘记录。

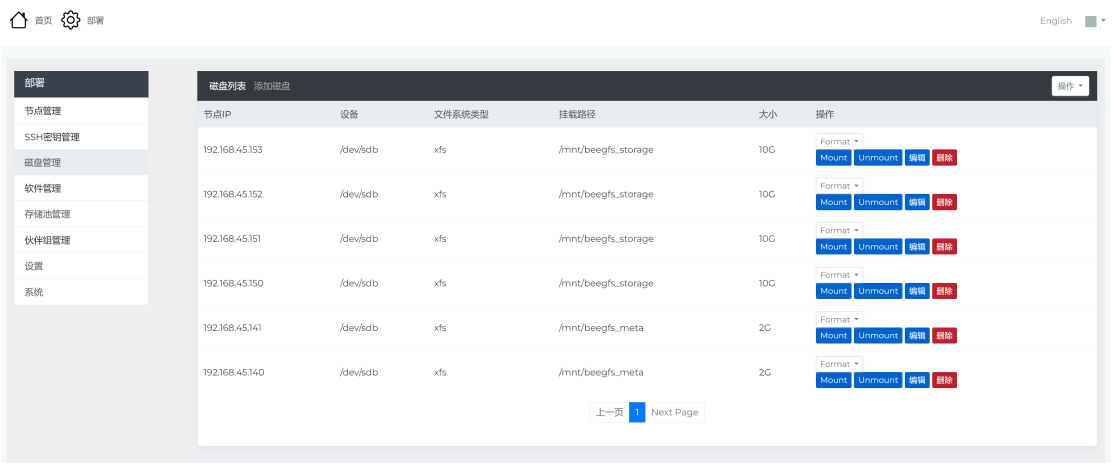
(5) 一旦所有的 Ansible 输出都被处理完毕，函数将重定向回磁盘页面，在前端显示磁盘扫描结果，如图 4-5(a) 所示。

此外，系统也提供了对单个磁盘进行操作的功能，包括添加、删除、格式化、挂载和卸载磁盘。添加操作需要通过提交表单并调用 Disk.add_disk() 函数实现，提交表单后，后端将磁盘信息录入数据库，并重定向回磁盘列表页面，用户即可查看添加结果。而删除、格式化、挂载和卸载磁盘调用的其他函数实现。用户在列表页点击按钮发起请求后，根据路由后端调用不同的函数，具体如下：

(1) 视图函数 delete_disk() 响应路由“/delete_disk”，它从请求的参数中获取磁盘 ID，后在数据库中删除对应的磁盘记录。

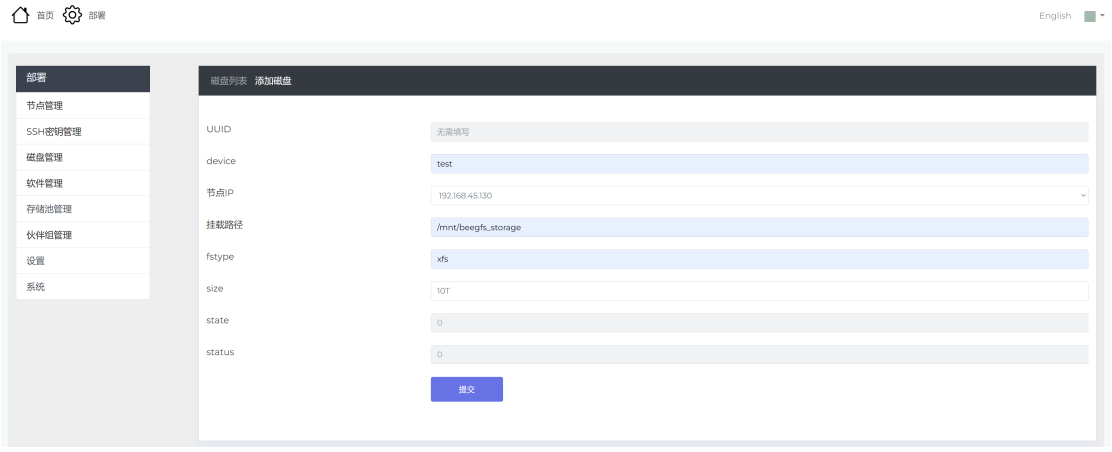
(2) 视图函数 format_disk() 响应路由“/format_disk”，它通过 Ansible 执行远程命令来格式化指定化磁盘，格式化成功后更新数据库中的磁盘状态和文件系统类型。

(3) 视图函数 mount_disk() 响应路由“/mount_disk”，它使用 Ansible 执行远



(a)　磁盘列表页面

(a)　List of disk



(b)　磁盘添加页面

(b)　Add page for disk

图 4-5　磁盘展示及管理页面

Figure 4-5　Page of disk display and management

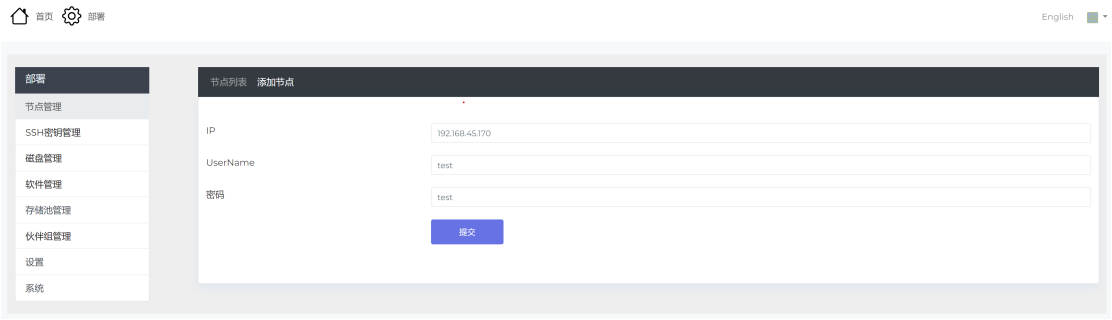
程挂载命令将磁盘挂载到用户提交表单时指定的目录，成功后更新数据库中的磁盘状态和挂载路径。

(4)　视图函数 unmount_disk_byid() 响应路由“/unmount_disk_byid”，它使用 Ansible 执行远程卸载命令以卸载指定磁盘，卸载成功后，更新数据库中的磁盘状态和清除挂载路径。

以上完成后，重定向到磁盘部署页面。这些函数结合了 Flask 的路由、表单处理、数据库操作以及 Ansible 的远程命令执行，共同实现了一个基本的磁盘管理系统。用户可以通过 Web 界面进行磁盘的添加、编辑、删除、格式化、挂载和卸载操作，而这些操作会反映在后端的数据库和实际的存储系统中。

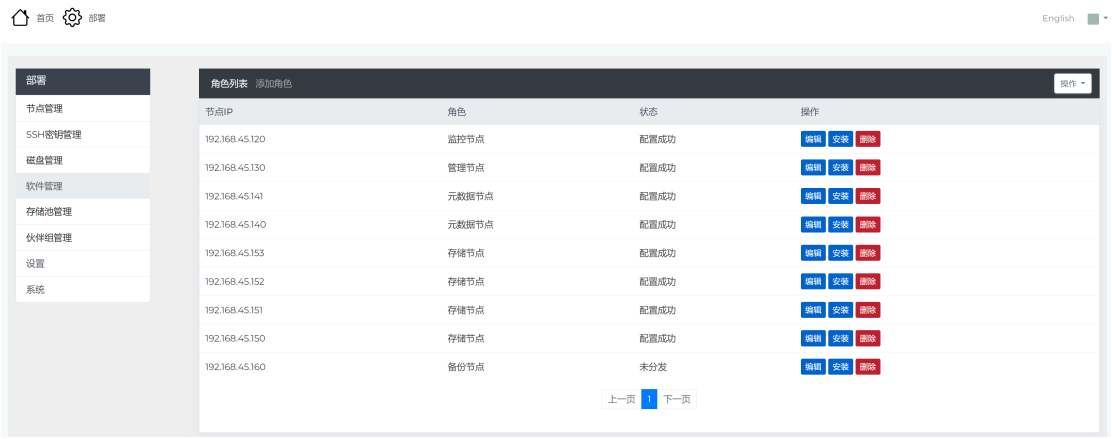
4.3.4 角色管理与部署

角色管理与部署功能是本模块的核心功能。本功能提供了一个 Web 界面以及对角色进行操作的功能接口，如图 4-6(a) 及 4-6(b) 所示。用户在列表页点击按钮发起请求后，后端根据路由调用不同的函数。



(a) 角色添加页面

(a) Add page for host_role



(b) 角色列表页面

(b) List of host_role

图 4-6 角色展示及管理页面

Figure 4-6 Page of host_role display and management

本模块提供两种安装方式：单个节点安装及批量节点统一安装，如图 4-7 所示。批量节点统一安装时为了便于集群初次部署时一次性安装多个不同种类节点。而单个节点安装适用于手动添加及部署单个节点的情况。因为在 Raspberry Pi OS 采用 Debian 操作系统及其衍生品的命令集，因此部署模块使用 Ansible 在远程调用命令时，也调用 Debian 操作系统命令集。此外为了统一各个服务的参数配置，将服务 id 均设置为 ip 最后一个字段。

1. 批量节点统一安装模式：

(1) 对于管理节点、元数据节点、存储节点、监控节点四个角色类型，执行以下操作：查询与该角色类型关联的所有 HostRole 记录，遍历每个 HostRole，并根据角色类型执行不同的操作。

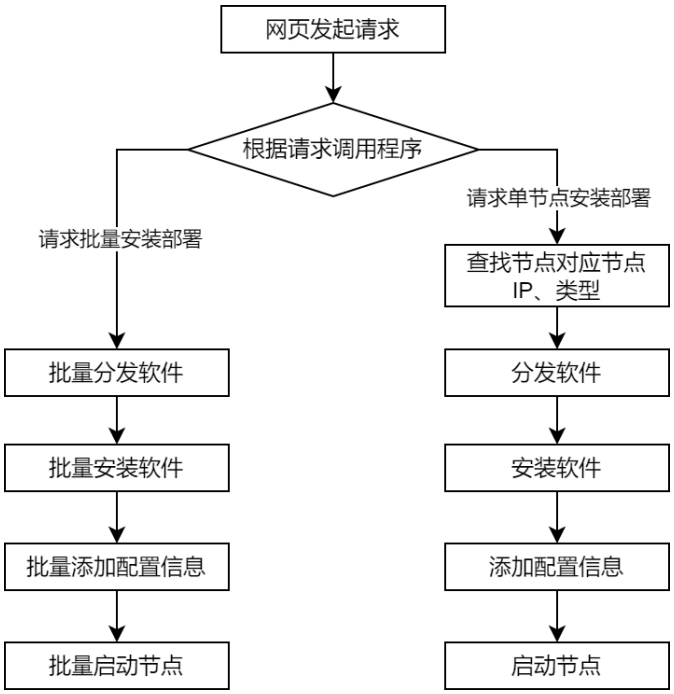


图 4-7　角色部署基本流程

Figure 4-7　Basic process of role deployment

(2) 调用 Ansible 的“copy”命令，指定源文件目录以及目标目录，将相应的软件安装包分发至各个节点。根据节点角色分发对应的软件包，包括：mgmtd 软件包、meta 软件包、storage 软件包、mon 软件包和 client 软件包。

(3) 确认软件包分发成功后，进入每个节点的相应软件包目录，使用 dpkg 命令安装相应的软件包。

(4) 在各个节点上使用命令配置节点信息，根据节点角色进行相应的配置。这可能涉及编辑配置文件、设置环境变量等操作。

(5) 确认软件配置成功后，启动软件。

(6) 确认以上步骤在每个节点上都成功完成后，将详细的安装结果传回。如果在执行过程中遇到错误，会更新相应 HostRole 的“status”字段，并继续处理列表中的下一个角色。

2. 单个节点安装模式：

(1) 用户在列表页点击按钮发起请求后，后端调用单节点软件安装部署函数，查询该节点对应的 HostRole，确定角色类型。

(2) 调用 rsync 命令将相应的软件安装包分发到该节点。

(3) 确认软件包分发成功后，进入节点的软件包目录，使用 dpkg 安装相应的软件包。

(4) 配置节点信息，并根据角色进行相应的配置。

(5) 确认软件配置成功后，启动软件。如果配置不成功，将相应的信息传回，并更新数据库中的“status”属性信息。

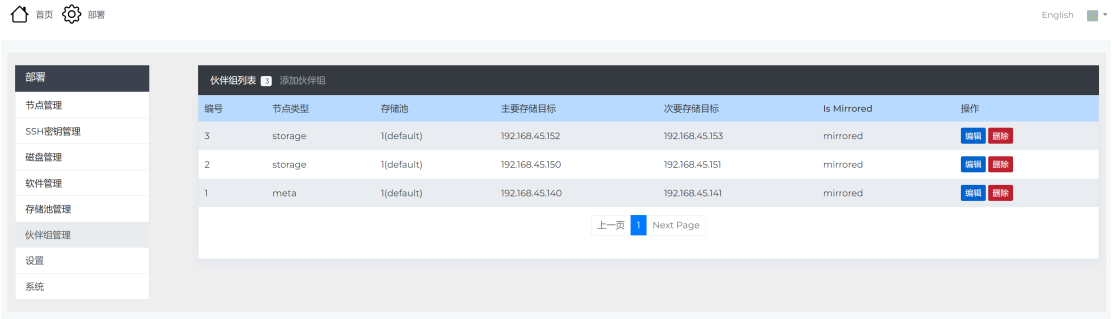
(6) 确认以上步骤在该节点上都成功完成后，将详细的安装结果传回。如果

在执行过程中遇到错误，会更新相应 HostRole 的“status”字段，并重定向回列表页面。

此外，系统也提供了对单个角色进行操作的功能，包括添加、删除、编辑角色。添加和修改操作需要通过提交表单并调用 HostRole.add_host_role() 函数实现，提交表单后，后端将角色信息录入数据库，并重定向回角色列表页面，用户即可查看添加结果。而删除通过视图函数 delete_host_role() 实现。它数响应路由“/delete_host_role”，它从请求的参数中获取响应角色 ID，后在数据库中删除对应的角色记录。

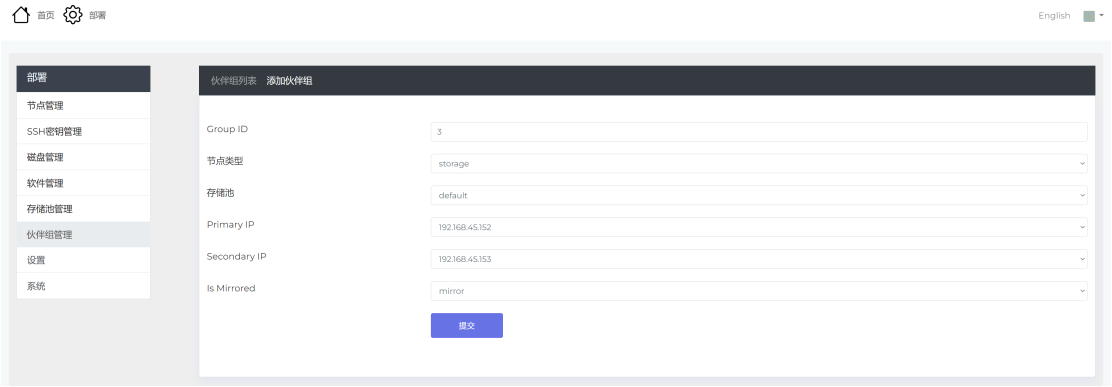
4.3.5 伙伴组管理

伙伴组是数据冗余和故障恢复的基础，对于分布式存储系统的管理非常重要。伙伴组管理功能提供了一个 Web 界面以及对伙伴组进行操作的功能接口，包括添加、删除节点，以及修改伙伴组信息，如图 4-8(a) 和 4-8(b) 所示。用户在列表页点击按钮发起请求后，后端根据路由调用不同的函数。



(a) 伙伴组列表页面

(a) List of buddygroup



(b) 伙伴组添加页面

(b) Add page for buddygroup

图 4-8 伙伴组展示及管理页面

Figure 4-8 Page of buddygroup display and management

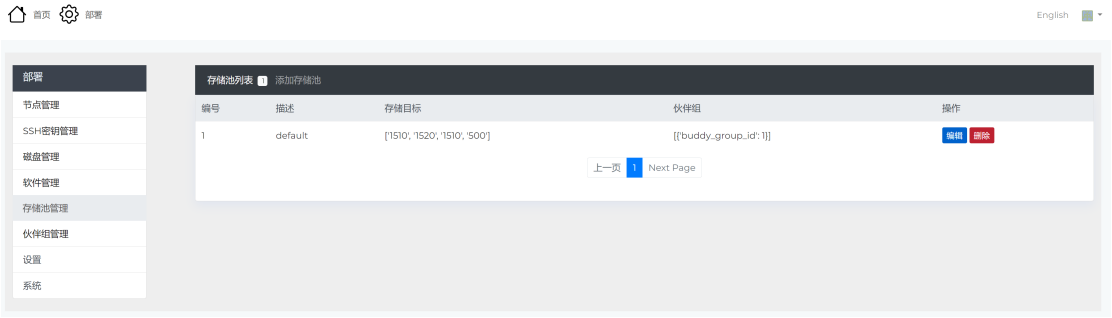
第一个视图函数 buddy_group() 响应“/buddy_group”路由的 GET 和 POST 请求。在 GET 请求中，它主要用于展示伙伴组列表或者添加伙伴组的表单。如

果是 GET 请求并且操作类型为“list”，则会显示伙伴组的分页列表，列表每行显示伙伴组的类型 ID、组 ID、存储池 ID、主 ID、副 ID 以及伙伴组是否被设置为镜像。如果不是列表操作，它将提供一个表单以便用户可以添加或编辑伙伴组。在 POST 请求中，它处理伙伴组的添加或编辑提交。处理器首先会检查操作类型，如果是添加操作，它会收集表单数据并创建一个新的伙伴组。如果是编辑操作，它将更新现有的伙伴组。无论是添加还是编辑操作，完成后都会重定向用户回到伙伴组列表页面。

第二个视图函数 delete_buddy_group() 响应路由“/delete_buddy_group”的 GET 请求，用于删除一个指定的伙伴组。它通过查询字符串获取伙伴组 ID，然后在数据库中找到并删除相应的记录。删除操作完成后，用户会被重定向回伙伴组列表页面。

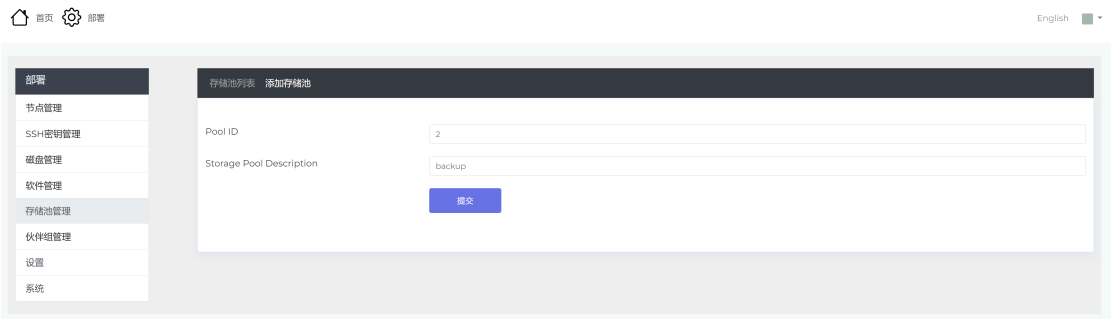
4.3.6　存储池管理

存储池管理实现了存储池的管理功能，存储池管理功能提供了一个 Web 界面以及对伙伴组进行操作的功能接口，包括添加、删除存储池，以及修改存储池名称，如图 4-9(a) 和 4-9(b) 所示。用户在列表页点击按钮发起请求后，后端根据路由调用不同的函数。



(a)　存储池列表页面

(a)　List of storage pool



(b)　存储池添加页面

(b)　Add page for storage pool

图 4-9　存储池展示及管理页面

Figure 4-9　Page of storage pool display and management

第一个视图函数 storage_pool() 响应路由“/storage_pool”的 GET 和 POST 请

求，并根据请求类型执行不同的操作。当用户想要查看存储池列表、添加或编辑存储池时，都会通过这个路函数。当函数接收到 GET 请求并且操作类型 “list” 时，则会显示伙伴组的分页列表。然后，函数会通过模板渲染这些信息，显示给用户。如果不是列表操作，它将提供一个表单以便用户可以添加或编辑存储池。当处理器接收到 POST 请求时，它首先会检查操作类型，如果是添加操作会调用 StoragePool.add_storage_pool() 方法来创建一个新的存储池。否则，会调用 StoragePool.edit_storage_pool() 方法来更新现有的存储池。无论是添加还是编辑，操作完成后，用户都会被重定向回存储池列表页面。

第二个视图函数 delete_storage_pool() 响应路由 “/delete_storage_pool” 的 GET 请求，用于删除指定的存储池。它通过请求的查询字符串参数获取存储池的 ID，然后在数据库中找到相应的记录并进行删除。完成数据库的变更提交后，用户会被重定向回存储池列表页面参数来显示。

4.4 自动化故障恢复模块实现

在实际部署中，用户可以根据对性能和数据冗余的需求选择合适的存储策略。BuddyMirror 模式允许存储服务在两个目标之一失败时仍可以访问所有数据，而 Raid0 模式具备更高的 IO 特性，但可用性依赖节点和磁盘的稳定性，两种模式各有优劣。本文将 BuddyMirror 模式用于数据备份，Raid0 模式用于射电数据的接收。本模块基于 BeeGFS 原有伙伴组的故障恢复功能，本节将详细介绍其实现。

4.4.1 故障检查及恢复流程

在 BeeGFS 原有的 BuddyMirror 模式中，实现了数据冗余但是没有实现节点替换和故障数据迁移，本文已在第三章中分析过：21CMA 存储系统设备数量庞大，故障难以避免，亟需全自动化的故障诊断和恢复能力。因此，本模块加入了备用节点替换机制，在 BeeGFS 的基础上实现了自动化节点替换和故障恢复能力，处理流程如图 4-10 所示。

BeeGFS 监控节点向元数据节点、存储节点发起消息请求，并将获取的节点状态信息保存到时序数据库 InfluxDB 中。本模块每隔一定时间从 InfluxDB 读取数据，通过遍历数据库中每个节点的信息来实时监控元数据节点、存储节点的可用性。若数据显示有节点存在异常，则调用 BeeGFS 的命令行控制工具检测该节点的应用程序运行情况，如果应用程序运行正常，则继续循环监测；如果服务运行异常，则对异常节点发起网络请求，检测节点的连通性，如果节点可以通过网络连接，则远程控制重启节点，否则将对硬件系统发起请求，通过电源管理系统控制节点重启。重启后，如果节点运行正常，则将重启操作记入故障恢复日志，并继续循环监测，否则判定节点异常。确认节点应用程序异常后，模块首先从数据库查询有无备用节点，若没有则记入故障恢复日志并在前端弹出提示，之后继续循环监测；若有备用节点，系统将检查是否有正在进行的数据读写操作，若无

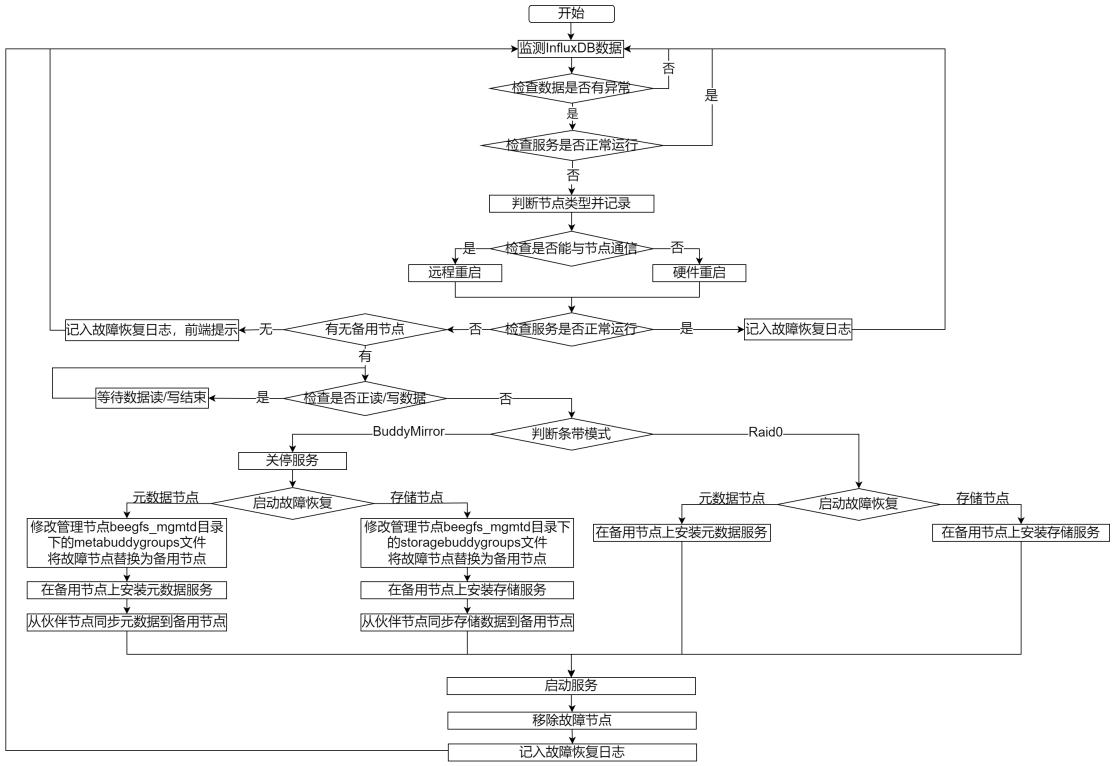


图 4-10　故障检查及恢复流程图

Figure 4-10　Flow chart of failure checking and recovery

数据读写，则关停服务，否则等待直至数据读写结束再关停服务。关停服务后启动故障恢复流程，不同条带模式采用不同策略。

4.4.2　节点替换及数据恢复功能

在原有 BeeGFS 开源的命令行工具 beegfs-ctl 中，并没有替换 BuddyGroup 中节点以及数据复制的功能。但是在 BeeGFS 的工作模式中，所有节点仅需在管理服务器注册，此后由管理节点指示其他节点该和哪些节点通信。所有关于节点注册信息及 BuddyGroup 的信息都保存在管理节点“/mnt/beegfs_mgmtd”目录中，因此如果能够适当的修改管理节点目录下的文件信息，并完整的将数据传输到新的节点中，即可实现自动化的故障恢复。由于管理节点有自带的守护进程，所以在其运行期间修改配置文件是无效的。为了能有效地修改管理节点保存的节点及配置信息，需要先将所有服务关闭后，再将管理服务关闭，之后再依次修改配置信息、复制数据、启动服务。下面是对节点替换和数据恢复功能的详细实现。

(1) BuddyMirror 模式

对于 BuddyMirror 模式，采用节点替换和数据迁移的方式。本系统采用一对树莓派构成 BuddyGroup，当一个元数据节点或存储节点故障时，另一节点的数据仍可访问。元数据节点的 BuddyGroup 信息保存在管理节点的配置文件 metabuddygroups 中，格式为“buddygroupID = primarynodeID , secondarynodeID”；存储

目标的 BuddyGroup 信息保存在配置文件 storagebuddygroups 中，格式为 “buddygroupID = primarytargetID , secondarytargetID”。接下来，模块根据监测 InfluxDB 数据时获取的节点 IP 和节点类型执行操作。首先，模块根据故障节点 IP 查询其相应的 BuddyGroupID。获得 BuddyGroupID 后，按照节点类型修改配置文件、复制数据。修改配置文件需要

对于元数据节点故障，模块根据故障节点和备用节点的 IP 获取对应 nodeID，修改管理节点目录 beegfs_mgmtd 下的配置文件 metabuddygroups，将对应 buddygroup 中的故障节点 nodeID 替换为备用节点 nodeID。修改完成后，在备用节点上安装元数据服务，并从伙伴节点的 beegfs_meta 目录复制 buddymir 文件夹到备用节点的 beegfs_meta 文件夹下。

对于存储节点故障，模块根据故障节点和备用节点的 IP 获取对应的 targetID，修改管理节点 beegfs_mgmtd 下的配置文件 storagebuddygroups，将对应 buddygroup 中的故障节点的存储目标 targetID 替换为备用节点的存储目标 targetID。修改完成后，在备用节点上安装存储服务，并从伙伴节点的 beegfs_storage 目录复制 buddymir 文件夹和 chunks 文件夹到备用节点的 beegfs_storage 文件夹下。

(2) RAID0 模式

对于 RAID0 模式，需要主机和磁盘保障其可用性。在本应用场景中，优先保障系统的在线可用以及最小的部署成本。当节点出现故障时，本模块将启动新的备用节点作为故障节点的替代，调用安装部署模块对存储系统进行快速重构，从而继续提供存储服务减少宕机时间。如果故障节点为元数据节点，则在备用节点上安装元数据服务；如果故障节点为存储节点，则在备用节点上安装存储服务。完成节点替换和数据迁移后，依次启动管理节点、元数据节点、存储节点，并从系统中移除故障节点，如果故障节点为存储节点则还需移除相应的存储目标。所有操作完成后，将故障恢复操作记入故障恢复日志，并继续循环监测。

4.5 单个集群监控模块实现

本节将详细描述单个集群监控模块关键数据提取方法以及数据可视化页面的实现。

4.5.1 集群关键数据提取

本文需从 InfluxDB 提取的信息只包括每个节点的运行情况及数据读写速率。节点的响应情况可以直接从表 meta 和表 storage 的 isResponding 字段得知，集群总 IO 速率和集群总存储容量及已用容量需要计算，其计算方法如算法 1 所示。

集群总 IO 速率为各个 RAID0 模式节点 IO 速率之和与各个 BuddyMirror 模式主节点 IO 之和；集群总存储容量为各个 RAID0 模式节点存储总容量与各个 BuddyMirror 模式主节点存储总容量之和，集群已用存储容量与各个 RAID0 模式节点已用容量和各个 BuddyMirror 模式主节点已用容量之和。计算集群总 IO 速率时，遍历所有节点，只将设置为 RAID0 模式或作为 BuddyMirror 的主节点

算法 1 Storage System Monitoring and Calculation

```
    procedure ClusterIO(node_data)
        total_io_rate ← 0
3:      for node in node_data do
            if node.mode == 'RAID0' OR node.is_primary_in_buddy_mirror then
                total_io_rate += node.io_rate
6:          end if
        end for
        return total_io_rate
9:  end procedure
    procedure ClusterStorage(node_data)
        total_storage ← 0
12:     used_storage ← 0
        for node in node_data do
            if node.mode == 'RAID0' OR node.is_primary_in_buddy_mirror then
15:             total_storage += node.total_storage
                used_storage += node.used_storage
            end if
18:     end for
        return total_storage, used_storage
    end procedure
```

的设备的 IO 速率累加起来，从而得到整个集群的总 IO 速率。计算集群总存储容量和已用存储容量的方法类似类似，遍历节点，累加符合条件的节点的总存储容量和已用存储容量，最后返回这两个值。

4.5.2　集群数据可视化

本模块通过读取和解析后端发送的存储区域状态数据，动态更新前端显示的状态和信息，使得用户能够实时看到存储系统的当前运行状态。最终形成的监控页面如图 4-11 所示，展示了单个存储设备的运行节点、备用节点、故障节点的数目、系统存储使用情况以及故障日志等信息。本部分使用 JavaScript 代码定义了一个函数“set_status”，它用于设置和更新存储区域的状态显示。函数接收一个包含区域数据的对象 area_data 作为参数，并从中读取多个属性以更新 UI 组件的状态。以下是函数的具体操作：

(1) 初始化变量：函数开始时初始化多个变量，包括 IP 地址、主机总数、正在运行的主机数、故障和备用主机数以及存储总量和已用存储量。

(2) 网页显示控制：根据集群状态，控制读写速率组件的显示与隐藏和存储容量的显示与否。如果集群状态是正常，网页显示集群完整信息。如果是初始状

态，节点数目会显示为“-”，表示还没有数据，并隐藏读写速率和存储容量组件。如果集群是故障状态，隐藏读写速率速率。

(3) 计算存储量显示格式：将以字节为单位的存储的总量和已用量，计算并转换为更直观的单位（M、G、T），以更直观的方式在用户界面上显示。

(4) 统计处在不同状态的节点数目，包括总节点数、正常运行节点数、备用节点数、故障节点数，并标记为不同的颜色。

(5) 显示集群的故障回复日志，每间隔一定时间将最近的几条故障恢复日志显示在前端。间隔时间和显示日志默认为 1min 和 4 条，可以根据需要修改。

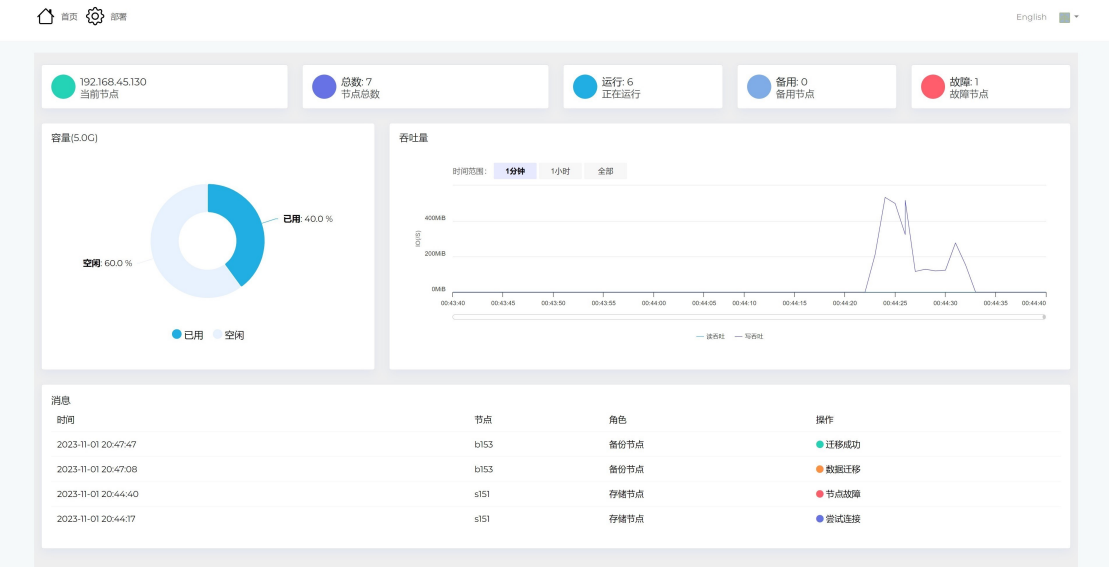


图 4-11　资源展示页面

Figure 4-11　Page of Resource display module

4.6　统一监控服务模块实现

21CMA 分布式存储系统最多可达 81 个存储设备，每个存储设备都是一套低功耗分布式存储系统及终端设备，需要实时跟进每个集群的运行状况。为了实时了解每个集群的运行状况，本文设计并实现了一个统一监控网站。通过与集群的监控服务器进行通信，系统能够获取每个存储集群的运行状况，并利用 HighCharts 图表对所有存储设备的状态进行定制化展示。

在单个集群部署网站的后端添加定时器，调用脚本实现数据传输。脚本通过 SQLAlchemy 每间隔 10s 读取集群状态数据表“Cluster”，并抽取 InfluxDB 中的 diskReadBytes（磁盘读速率），diskWriteBytes（磁盘写速率）。脚本通过 Psycopg2 包将数据传输至统一监控服务器。统一监控服务系统设计部署一个 PostgreSQL 数据库接收记录每个存储设备的运行情况，具体流程图流程如图 4-12 所示。

根据 status 将集群状态分为四种：初始状态、正常、故障及恢复中。初始状态指的是集群已注册但未开始运行，正常状态指集群已注册且正常读写数据，故

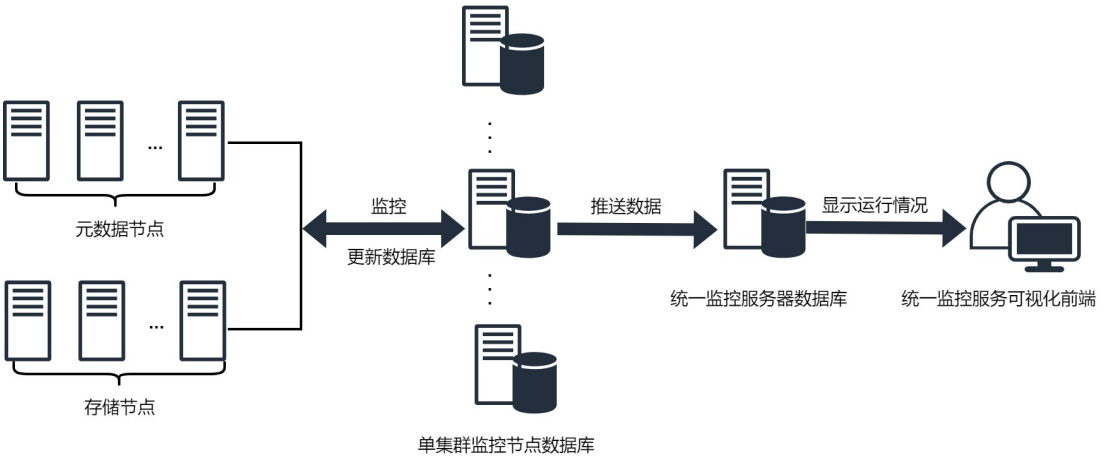


图 4-12　存储设备监控数据

Figure 4-12　Monitoring data

障状态指节点发生故障无法运行，而恢复中状态则表示节点正在执行故障恢复流程。页面包含多个仪表盘，每个仪表盘对应一个集群的运行状况，包括运行集群节点总数、运行节点数目、故障节点数目、备用节点数目，以及集群 IO 速率和集群容量占用，每个仪表盘可视为单个集群监控页面的精简，便于展示集群运行及使用情况，具体实现方式如算法 2 所示。

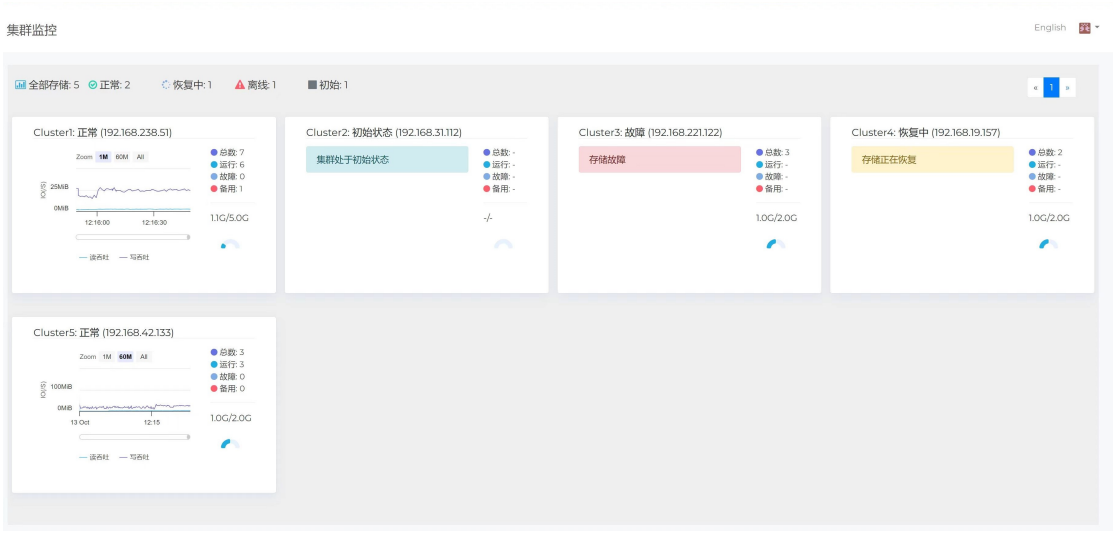


图 4-13　存储设备的统一监控及可视化

Figure 4-13　Unified monitoring and visualization of storage devices

统一监控服务后端通过检索数据库表提取相应的数据，进而在前端展示集群运行情况。系统对多个存储设备进行数据抽取，最终形成统一的监控页面。监控页面如 4-13 所示。

算法 2 资源监控状态更新

```
1:  procedure UpdateResourceMonitor
2:      Initialize
3:      StorageTotal ← ConvertStorageUnit(TotalStorage)
4:      StorageUsed ← ConvertStorageUnit(UsedStorage)
5:      if ClusterStatus ==′ Normal′ then
6:          SetStorageStatusText(green,"Storageishealthy")
7:      else if ClusterStatus ==′ Failure′ then
8:          SetStorageStatusText(red,"Storageisinfailurestate")
9:      else
10:         SetStorageStatusText("−","Initializing...")
11:     end if
12:     UpdateHostsStatus(TotalHosts, RunningHosts, FailedHosts, StandbyHosts)
13:     if ClusterStatus ==′ Normal′ then
14:         ShowReadWriteRateComponent()
15:         ShowStorageCapacity(StorageTotal, StorageUsed)
16:     else if ClusterStatus ==′ Failure′ then
17:         HideReadWriteRateComponent()
18:         ShowStorageCapacity(StorageTotal, StorageUsed)
19:     else
20:         HideReadWriteRateComponent()
21:         HideStorageCapacity()
22:     end if
23: end procedure
```

4.7 小结

本章对系统进行了详细设计与实现。根据第三章的设计原则，详细设计并实现了系统的数据库，并实现了系统的 Web 框架及各个功能模块。部署与软件管理模块通过配置管理工具实现了软硬件的一键部署、灵活增加或删除容量和灵活升级。故障恢复模块通过管理节点的配置文件以及故障节点到备用节点的数据复制，对伙伴组进行了进一步的管理和维护，从而提升了系统的可靠性。监控模块分为单个集群监控模块和统一监控模块，共同实现了集群信息的可视化。图表定制功能提高了整体的用户体验。统一监控网站的改进之处在于其能够自动适应集群数量，具备灵活性和可扩展性。通过模块化的设计，系统能够轻松适应 21CMA 分布式存储系统的未来扩容需求，确保长期稳定运行。

第 5 章　系统应用及测试

本章将介绍系统的应用方法以及系统的测试过程，包括性能测试的结果分析，目的是验证系统是否满足 21CMA 存储集群的管理需求、可靠性需求和效率需求。

5.1　系统应用

本系统包括部署及软硬件管理模块、故障恢复模块以及统一监控网站。部署及软硬件管理模块部署在每个集群的监控节点上，统一监控网站部署在统一监控服务器上。

5.1.1　单个集群运维管理系统部署及应用

1. 安装 python 环境，以便在集群监控节点上运行系统。执行以下命令来安装 Python 3.6.8：

(1) conda search python

(2) conda install python=3.6.8

2. 安装项目所需的所有 Python 库和依赖项。通过以下命令来安装：

(1) pipenv install - -skip-lock

3. 安装 Ansible，以便自动化管理和配置集群节点。通过以下命令在本地安装 Ansible：

(1) pip install ansible

3. 在本地初始化数据库、执行数据库迁移、进行系统初始化，并启动系统服务。通过以下命令来启动系统：

(1) flask db init

(2) flask db migrate

(3) flask db upgrade

(4) flask init

(5) flask run

5.1.2　统一监控网站部署及应用

1. 安装 python 环境，以便在统一监控服务器上运行系统。执行以下命令来安装 Python 3.6.8：

(1) conda search python

(2) conda install python=3.6.8

2. 安装 PostgreSQL 数据库，用于存储统一监控系统的数据。通过以下命令来安装 Postgres：

(1) sudo apt install postgresql postgresql-contrib

3. 在本地初始化数据库、执行数据库迁移、进行系统初始化，并启动系统服务。通过以下命令来启动系统：

(1) flask db init
(2) flask db migrate
(3) flask db upgrade
(4) flask init
(5) flask run

5.2 系统测试

本节为部署及软硬件管理模块、故障恢复模块、统一监控服务模块测试。

5.2.1 测试环境

为了实际测试本文中构建的分布式存储系统运维管理系统的功能，本文使用 Vmware 虚拟机作为测试节点。每个节点的基本配置为：

- CPU：Intel(R) Core(TM) i7-8550U CPU 2 Core @ 1.80 GHz
- 内存：2 GB
- 硬盘 1：10GB
- 操作系统：Ubuntu 18.04.4 LTS

而元数据节点和存储服务节点额外添加了硬盘 2，元数据节点硬盘 2 大小为 2GB，存储服务器硬盘 2 大小为 10GB。

5.2.2 部署及软硬件管理模块测试

Playwright 是一个开源的自动化测试框架，可用于 Web 应用的端到端测试，它提供了一套简洁的 API 来模拟用户的各种交互行为，如点击、输入、文件上传等。本文使用 PlayWright 进行交互测试，检查部署及软硬件管理、自动化故障恢复模块能否按照预期工作。

1. 本部分为对节点管理、磁盘管理、存储池管理、伙伴组管理功能的测试，方式为对使用 PlayWright API 测试从用户提交请求到重定向回列表列表的时间。测试次数为每种功能测试 100 次，其结果如表 5-1 所示。

对节点、磁盘、存储池、伙伴组的增加、删除、修改操作所需时间均在 400ms 以下，时间较为接近，即表明将信息录入数据库、网页加载两个步骤加起来的时间控制在 400ms 以内。

2. 本部分为对单节点密钥分发、扫描磁盘、软件部署、软件启动的测试，方式为使用 PlayWright API 测试从用户提交请求到重定向回列表列表的时间。通过这种方式可以系统是否成功对远程节点进行操作。测试次数为每种功能测试 10 次，其结果如表 5-2 所示。

表 5-1　软硬件操作时间及成功率

Table 5-1　Time and success rate of hardware and software operations

测试场景	操作	完成时间/ms	成功率/%	操作	完成时间/ms	成功率/%	操作	完成时间/ms	成功率/%
节点操作	增加	≤ 400	≥ 99	删除	≤ 400	≥ 99	修改	≤ 400	≥ 99
磁盘操作	增加	≤ 400	≥ 99	删除	≤ 400	≥ 99	修改	≤ 400	≥ 99
存储池操作	增加	≤ 400	≥ 99	删除	≤ 400	≥ 99	修改	≤ 400	≥ 99
伙伴组操作	增加	≤ 400	≥ 99	删除	≤ 400	≥ 99	修改	≤ 400	≥ 99

表 5-2　单节点部署操作时间及成功率

Table 5-2　Time and success rate of single-node deployment operations

测试场景	完成时间/ms	成功率/%
密钥分发	≤ 400	≥ 99
扫描磁盘	≤ 400	≥ 99
部署软件	≤ 400	≥ 99
启动软件	≤ 400	≥ 99

单节点密钥分发、扫描磁盘、软件部署、软件启动所需的时间均在 400ms 以下，时间较为接近，即表明实际后端操作所需时间极少，时间仍主要用于将信息录入数据库、网页加载，两个步骤加起来的时间控制在 400ms 以内。

5.2.3　故障恢复模块测试

本小节分别对存储节点故障和元数据节点故障进行模拟，对 BuddyMirror 和 Raid0 两种模式下的故障恢复能力进行了评估。每种故障场景模拟执行 10 次得到了相应的结果。测试结果如表 5-3 所示，在 BuddyMirror 模式下，系统能够在 50s 内完成数据初始化及启动动作，数据安全性得到了提升。而在 Raid0 模式下，系统可在 10s 内完成文件系统重构，极大地减少了系统的故障时间。

表 5-3　故障恢复时间及成功率

Table 5-3　Runtime and success rate of the selfrecovery

测试场景	完成时间/ms	成功率/%
存储服务器 BuddyMirror 模式	≤ 50000	≥ 90
存储服务器 Raid0 模式	≤ 9800	≥ 90
元数据服务 BuddyMirror 模式	≤ 29000	≥ 90
元数据 Raid0 模式	≤ 9800	≥ 90

5.2.4 统一监控模块测试

本小节通过脚本模拟监控服务器同时监控 81 个集群的情况，并监控数据查询速率。测试结果如图 5-1 所示，数据库查询平均速度保持在 0.1s 以内，在运行 1.5h 后出现过卡顿但整体运行速率未受影响，稳定性良好，说明集群监控系统能够及时检测每个集群的潜在问题。

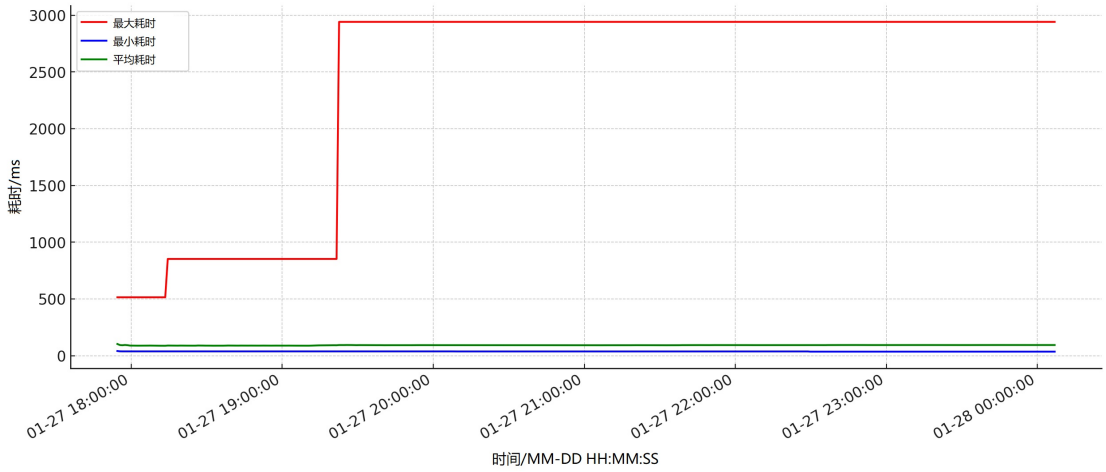


图 5-1 统一监控服务数据库查询耗时

Figure 5-1 Time consumed by unified monitoring service database queries

5.3 小结

本章介绍了系统的应用方法，分析了系统的测试结果。本章设置了实际的测试环境并设计了测试方案，采用自动化测试工具 Playwright 对系统进行了多方面的测试。测试结果显示，系统在多个方面均展现出良好的性能和稳定性，满足了 21CMA 存储集群管理的可靠性与效率要求。本章的工作为系统提供了强有力的实证支持，证明了系统设计和实现是成功和有效的。

第 6 章　总结

6.1　结论与创新点

随着阵列数目增多，大型射电望远镜阵列的观测数据呈现爆炸式的增长。如何科学地处理和存储这些海量数据，是射电望远镜普遍面对的问题。目前看来，对天文数据进行分级并采取分布式存储是有效解决射电望远镜数据存储的首选方法。

本文从 21CMA 的存储需求出发，针对当前天文观测数据存储与管理的挑战，提出了一系列创新的解决方案。本文在分布式存储系统的自动化部署、故障恢复以及监控可视化方面做出了大胆尝试，本文主要包括以下成果：

1. 越来越多的低频阵列望远镜开始采用分布式存储方案，低频阵列望远镜通常有多个数据存储设备。本文提出了创新的低频阵列望远镜数据存储设备部署方案。部署及软硬件管理模块通过配置管理工具 Ansible 实现了自动化部署流程及软硬件管理功能，有效解决了 21CMA 集群设备数量庞大导致的手动部署费时费力、以及集群节点配置一致性的问题。

2. 射电望远镜已有采用 BeeGFS 做数据接收与存储系统的先例，但目前尚未有对 BeeGFS 的针对性改进。自动化故障恢复模块在原有 BeeGFS 故障恢复功能的基础上进行了扩展，实现了故障节点替换和数据迁移，使得 21CMA 数据存储集群可以在无人值守的情况下实现故障恢复，提升了系统的稳定性和可靠性。

3. 射电望远镜资源调度需要对集群的存储容量、硬件状况、集群运维日志的记录，而原有的 BeeGFS 监控功能缺乏对这些信息的汇总和展示。本文通过集中的监控数据可视化页面综合展示了单个集群的节点运行情况、读写性能、集群使用情况以及故障恢复日志。

4. 低频阵列望远镜通常由多个存储设备，但分布式存储系统的监控功能缺乏对多个集群运行状态的收集和统一展示。统一监控服务页面提供了对所有集群的实时视图，从而使运维团队能够迅速响应潜在的系统警报和性能问题，把握系统整体维修量变化趋势，合理分配和调度资源。

综上，本文构建了一套运维管理系统，能够有效地管理集群的软硬件资源、高效地监控集群状态、实现集群自动化故障恢复以保证系统稳定运行。本文对于类似的低频阵列望远镜具有重要的参考价值。

6.2　不足与展望

本文的研究内容有创新性和实际应用价值，为未来类似的大规模阵列望远镜科研项目提供了实践经验。同时，本文还可以在以下方面做进一步改进：

1. 系统目前仅针对基于 BeeGFS 的集群进行了优化和测试，但缺乏对于其

他类型的分布式存储系统的支持。在未来的工作中，可以进一步提升系统的通用性，实现对更多类型分布式存储系统的支持。

2. 故障恢复机制虽然可以自动化地诊断和处理大部分故障，但对于某些复杂故障的根源分析和预防措施还需要进一步加强。在未来的工作中，可以加强故障根源分析和预防机制，进一步增强容错性。

3. 受到条件限制，本文只利用虚拟机进行了测试，缺乏在树莓派上的测试，今后将进行更充分的测试和分析。

未来，本运维管理系统有望发展成为一个更加通用、智能化和用户友好的分布式存储管理平台，为科研数据的存储和处理提供坚实的支撑。

参考文献

Guo S, Lu Y, An T, 等. 面向 SKA1 时代的科学数据流及阵列模拟分析 [J/OL]. SCIENTIA SINICA Physica, Mechanica & Astronomica, 2023, 53(2): 229504-. http://www.sciengine.com/publisher/ScienceChinaPress/journal/SCIENTIASINICAPhysica,Mechanica&Astronomica/53/2/10.1360/SSPMA-2022-0261.

武向平. 中国 SKA 科学报告 [M]. 北京: 科学出版社, 2019.

刘奇, 刘晔, 陈卯蒸, 等. NSRT 台站 L 频段 RFI 测量及干扰缓解 (英文) [J]. 天文学报, 2019, 60 (6): 82-95.

张海龙, 朱艳, 聂俊, 等. 新疆天文台 NSRT 观测数据存储系统 [J]. 2018.

张海龙, 裴鑫, 聂俊, 等. 110 米射电望远镜项目信息技术挑战 [J]. 2018.

张萌, 张海龙, 王杰, 等. 应用于射电天文的高效实时管道数据流传输与处理技术 [J]. 2021.

徐怡冬, 张鑫. SKA 与 21cm 宇宙学 [J]. 中国科学: 物理学力学天文学, 2020, 50(07): 12-17.

杨传辉. 原理解析与架构实战 [M]. 机械工业出版社, 2013: 293.

王娜. 新疆奇台 110 米射电望远镜 [J]. 中国科学: 物理学, 力学, 天文学, 2014(8): 783-794.

韩军, 朱炜玮, 岳友岭. 低功耗分布式存储系统及终端设备: CN113630441B [P]. 2023.

Abramson D, Jin C, Luong J, et al. A beegfs-based caching file system for data-intensive parallel computing [C]//Asian Conference on Supercomputing Frontiers. Springer International Publishing Cham, 2020: 3-22.

BeeGFS. Performance evaluation of the beegfs file system on the arm aarch64 architecture [EB/OL]. 2022. https://thinkparq.com/wp-content/uploads/2019/08/White_Paper_BeeGFS_File_System-_on_the_Arm-_AArch64-_Architecture.pdf.

Belikov A, Boxhoorn D, Dijkstra F, et al. Target for lofar long term archive: Architecture and implementation [J]. arXiv preprint arXiv:1111.6443, 2011.

Berkhout L M, Jacobs D C, Abdurashidova Z, et al. Hydrogen epoch of reionization array (hera) phase ii deployment and commissioning [J]. arXiv preprint arXiv:2401.04304, 2024.

Bertocco S, Goz D, Tornatore L, et al. Inaf trieste astronomical observatory information technology framework [J]. Astronomical Data Analysis Software and Systems XXIX, 2020, 527: 303.

Boyer E B, Broomfield M C, Perrotti T A. Glusterfs one storage server to rule them all [R]. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2012.

Braam P. The lustre storage architecture [J]. arXiv preprint arXiv:1903.01955, 2019.

Brown R L, Wild W, Cunningham C. Alma–the atacama large millimeter array [J]. Advances in Space Research, 2004, 34(3): 555-559.

Carilli C, Rawlings S. Science with the square kilometer array: Motivation, key science projects, standards and assumptions [J]. arXiv preprint astro-ph/0409274, 2004.

Carlson M, Yoder A, Schoeb L, et al. Software defined storage [J]. Storage Networking Industry Assoc. working draft, 2014: 20-24.

DeBoer D R, Parsons A R, Aguirre J E, et al. Hydrogen epoch of reionization array (hera) [J]. Publications of the Astronomical Society of the Pacific, 2017, 129(974): 045001.

Gudu D, Hardt M. Arm cluster for performant and energy-efficient storage [J]. Computational Sustainability, 2016: 265-276.

Guzman J C, Chapman J, Marquarding M, et al. Status report of the end-to-end askap software system: towards early science operations [J]. Software and Cyberinfrastructure for Astronomy IV, 2016, 9913: 991311.

Heichler J. An introduction to beegfs [J]. Introductionto BeeGFS by ThinkParQ. pdf, 2014.

Holties H, Renting A, Grange Y. The lofar long-term archive: e-infrastructure on petabyte scale [C]//Proc. of SPIE Vol: volume 8451. 845117-1.

Jongerius R, Wijnholds S, Nijboer R, et al. An end-to-end computing model for the square kilometre array [J]. Computer, 2014, 47(9): 48-54.

MacMahon D H, Price D C, Lebofsky M, et al. The breakthrough listen search for intelligent life: a wideband data recorder system for the robert c. byrd green bank telescope [J]. Publications of the Astronomical Society of the Pacific, 2018, 130(986): 044502.

Napier P J, Thompson A R, Ekers R D. The very large array: Design and performance of a modern synthesis radio telescope [J]. Proceedings of the IEEE, 1983, 71(11): 1295-1320.

Ord S, Tremblay S, McSweeney S, et al. Mwa tied-array processing i: Calibration and beamformation [J]. Publications of the Astronomical Society of Australia, 2019, 36: e030.

Padmanabh P, Barr E, Sridhar S, et al. The mpifr–meerkat galactic plane survey–i. system set-up and early results [J]. Monthly Notices of the Royal Astronomical Society, 2023, 524(1): 1291-1315.

Pedretti K, Younge A J, Hammond S D, et al. Chronicles of astra: Challenges and lessons from the first petascale arm supercomputer [C]//SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020: 1-14.

Price D C, MacMahon D H, Lebofsky M, et al. The breakthrough listen search for intelligent life: Wide-bandwidth digital instrumentation for the csiro parkes 64-m telescope [J]. Publications of the Astronomical Society of Australia, 2018, 35: e041.

Renting G, Holties H. Lofar long term archive [J]. Astronomical Data Analysis Software and Systems XXI, 2012, 461: 689.

Sanidas S, Cooper S, Bassa C, et al. The lofar tied-array all-sky survey (lotaas): Survey overview and initial pulsar discoveries [J]. Astronomy & Astrophysics, 2019, 626: A104.

Schaubert D, Boryssenko A, Van Ardenne A, et al. The square kilometer array (ska) antenna [C]// IEEE International Symposium on Phased Array Systems and Technology, 2003. IEEE, 2003: 351-358.

Schuh H, Behrend D. Vlbi: A fascinating technique for geodesy and astrometry [J]. Journal of geodynamics, 2012, 61: 68-80.

Singh S. Big bang: The origin of the universe [M]. p. 406: Harper Perrennial, 2005: 402-408.

Weil S, Brandt S A, Miller E L, et al. Ceph: A scalable, high-performance distributed file system [C]// Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI'06). 2006: 307-320.

Wu C, Wicenec A, Pallot D, et al. Optimising ngas for the mwa archive [J]. Experimental Astronomy, 2013, 3(36): 679-694.

Xue M, Bhat N, Tremblay S, et al. A census of southern pulsars at 185 mhz [J]. Publications of the Astronomical Society of Australia, 2017, 34: e070.

Zheng Q, Wu X P, Johnston-Hollitt M, et al. Radio sources in the ncp region observed with the 21 centimeter array [J]. The Astrophysical Journal, 2016, 832(2): 190.

致 谢

硕士生涯即将结束，在此，我诚挚地感谢所有支持和帮助过我完成本论文的人。

首先，我要特别感谢我的导师崔辰州研究员和韩军副研究员。崔老师在学术上拥有丰富的经验，在生活中和蔼可亲，给我提供了宝贵的指导和热心的支持。韩老师待人随和，耐心地协助我解决在研究中遇到的各种问题。没有他们的指导和帮助，我无法完成这篇论文。

还要感谢国家天文台天文信息技术团组的所有成员，他们是樊东卫、何勃亮、李长华、李珊珊、米琳莹、陶一寒、王有芬、许允飞、杨涵溪、杨丝丝老师，非常荣幸能够在组里度过硕士期间的科研时光。

感谢我的同学和师弟师妹，他们是吴莹、张琦乾、张震、马鹏辉、邵务俊、朱珈莹、左肖雄、陈朗、汤超，他们在学术交流和日常生活中给予了我很大的帮助和激励。非常有幸能和大家成为同学。

感谢我的室友蔡楠楠、姜雨儿、吴仪。我们共同度过的三年充满了快乐和支持，她们让我在学习和生活上都更加坚定自信。非常有幸能和她们成为室友和朋友。

感谢国台教育处的梁艳春老师、李响老师、马怀宇老师、毛永娜老师、郑菲菲老师，他们无私的工作确保了我们学术活动的顺利进行。

还要感谢评阅我的论文和参与答辩的所有老师，感谢他们在百忙之中抽出时间评阅我的拙作，给我的论文提出指导意见，参与我的论文答辩。

最后，感谢爸爸妈妈、弟弟、我的闺蜜、东岳，他们一直是我最坚实的后盾，在我遇到困难时给予了我无限的支持和鼓励。

再次感谢所有帮助和支持过我的人，祝愿大家健健康康开开心心！

2024 年 6 月

作者简历及攻读学位期间发表的学术论文与其他相关学术成果

作者简历:

2016 年 9 月——2020 年 6 月，在西北大学信息科学与技术学院获得学士学位。

2020 年 9 月——2024 年 6 月，在中国科学院大学天文与空间科学学院获得硕士学位。

已发表（或正式接受）的学术论文:

(1) 杨嘉宁，韩军，崔辰州. 面向 21CMA 的分布式存储运维管理系统设计与实现. 中国科学院大学学报，2024（已接收）.

申请或已获得的专利:

发明专利：杨嘉宁，韩军，崔辰州；低功耗并行存储运维管理系统；CN 117539724 A（进入实质审查阶段）

参加的研究项目及获奖情况:

参与科技部 SKA 专项《SKA 脉冲星搜寻预研》项目