分类号：P208　　单位代码：10335

密　级：____　　学　　号：21738054

浙江大学

硕士学位论文

中文论文题目： **面向分布式存储系统 Ceph 的遥感影像瓦片存储及其关键技术**

英文论文题目： **Remote sensing image tile storage and its key technology for distributed storage system Ceph**

申请人姓名：曹晓裴

指导教师：杜震洪 教授

专业名称：地质工程

研究方向：地理大数据存储

所在院系：地球科学学院

论文提交日期　2020 年 6 月 22 日

面向分布式存储系统 Ceph 的遥感影像瓦片存储及其关键技术

论文作者签名：曹晓裴

指导教师签名：

论文评阅人 1：隐名评阅

评阅人 2：隐名评阅

评阅人 3：隐名评阅

评阅人 4：

评阅人 5：

答辩委员会主席：钱沄涛\教授\浙江大学

委员 1：钱沄涛\教授\浙江大学

委员 2：周斌\教授\杭州师范大学

委员 3：刘仁义\教授\浙江大学

委员 4：杜震洪\教授\浙江大学

委员 5：张丰\副教授\浙江大学

答辩日期：2020 年 6 月 10 日

Remote sensing image tile storage and its key technology for distributed storage system Ceph

Author's signature: Cao Xiaopei

Supervisor' s signature: Zhenhong Du

External Reviewers: Double-Blind Peer Review

Double-Blind Peer Review

Double-Blind Peer Review

Examining Committee Chairperson:

Qian Yuntao\Professor\Zhejiang University

Examining Committee Members:

Qian Yuntao\Professor\Zhejiang University

Zhou Bin\Professor\Hangzhou Normal University

Liu Renyi\Professor\Zhejiang University

Du Zhenhong\Professor\Zhejiang University

Zhang Feng\ Associate Professor\Zhejiang University

Date of oral defence: June 10,2020

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 浙江大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：曹晓斐　　签字日期：2020 年 6 月 22 日

学位论文版权使用授权书

本学位论文作者完全了解 浙江大学 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 浙江大学 可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：曹晓斐　　导师签名：

签字日期：2020 年 6 月 22日　　签字日期：2020 年 6 月 22日

致谢

时光如水，岁月如歌。不知不觉间，我已站在毕业的十字路口，即将告别充实完美的研究生求学生活。回想三年前，我怀着满腔的学术热情，带着些许的迷茫机缘巧合地来到实验室，期间充实的科研生活使我从一问三不知的技术小白逐渐成长为热爱技术的程序员。白驹过隙，转眼就要离开陪伴了我三年的老师和同学们，心中不禁感慨万千。感谢实验室给我带来的机遇与挑战，让我变得更加强大，更要感谢一直相信并鼓励我的朋友和家人，助我跨越一道道难关。

值此毕业论文完成之际，由衷地感谢实验室的刘仁义老师、张丰老师和杜震洪老师，你们在项目工作中对我高标准严要求的指导，让我在团队中快速成长。刘老师“狼群作战”的理念让我印象深刻，您对待工作的认真与拼劲也让我钦佩不已。张老师“少说话多做事”的作风深深影响了我，您虽然不苟言笑但是对学生的严格要求让我终生受益。杜老师的幽默风趣为我忙碌的实验室生活增添了不少欢声笑语。还要感谢学院的谭超老师、朱建丽老师和姜俊辉老师在学习上给予我的帮助与关心，让我倍感温暖。

感谢实验室的师兄师姐、同学以及师弟师妹们，和你们相处的这段时光让我更加清楚自己想要成为什么样的人。感谢师兄们在项目工作中对我的鼓励和鞭策，让我学会了独立思考。感谢师姐们在科研生活中对我无微不至的关怀，让我在浙大体会到人性的温暖。感谢同窗三年的小伙伴在学习和生活中给予我的帮助，与你们共同拼搏的日子将使我终生难忘。感谢师弟师妹们在项目中为我分担了很多压力，在我身心疲惫之时给我带来满满的正能量。

特别感谢我的室友梁钢，在我最无助的时候陪我哭、最开心的时候陪我笑，孝顺懂事的你一直是我学习的榜样。特别感谢我的女朋友黄婕，遇见你是我的幸运，你的陪伴与支持帮我走出人生中一个又一个低谷，优秀的你也激励我成为更好的人。特别感谢我最爱的爸爸妈妈一直以来的陪伴与鼓励，感恩有你们。

曹晓裴

二零二零年六月

于浙江杭州

摘要

近年来，随着遥感技术以及信息技术的快速发展，遥感影像已经成为科学研究、行业应用以及互联网企业最重要的地理数据之一。此外，为了有效提高遥感影像数据的快速共享、处理、分析与显示效率，影像切片技术逐渐成为影像对外提供服务的重要方式之一，因此，遥感影像瓦片也成为各行业部门对外提供服务的基础数据，如何对海量遥感影像瓦片进行高效组织、存储和管理是遥感影像对外提供高性能服务的关键所在。传统的基于数据库和文件系统的遥感影像瓦片存储系统，不仅在存储性能方面表现较差，同时也难以扩展和维护，已经无法满足海量遥感影像瓦片的存储。而新一代的基于主从架构的NoSQL数据库以及HDFS分布式文件系统，由于存在主节点单点故障问题，其主节点往往容易成为整个存储系统的性能瓶颈。此时，基于无中心化架构思想的分布式存储系统Ceph由于其被设计成一款没有单点故障，具有高性能、高可靠性以及高扩展性的系统，已经成为海量遥感影像瓦片存储的最佳选择。但其在存储海量遥感影像瓦片这样的小文件时具有明显的性能缺陷。针对上述问题，本文在前人的小文件合并方案的基础上，充分考虑了遥感影像瓦片的时空特性，设计并实现了海量遥感影像瓦片多级优化存储方法TMOSM。本文主要的研究内容如下：

（1）设计了海量遥感影像瓦片多级优化存储方法TMOSM。针对Ceph在存储海量遥感影像瓦片等小文件时性能表现不佳的问题，本文基于遥感影像瓦片合并和预取缓存策略设计了海量遥感影像瓦片多级优化存储方法TMOSM。从遥感影像瓦片合并存储策略、全局映射索引策略以及预取缓存策略等多方面对面向Ceph分布式存储系统的遥感影像瓦片存储进行了读写性能优化。

（2）提出了基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略。为了将具有高度空间相关性的遥感影像瓦片合并在一起，同时考虑合并服务集群的伸缩性，本文通过分析已有数据划分方法的不足和缺陷，提出了基于扩展Z曲线和一致性哈希的遥感影像瓦片合并策略，该策略能够将空间上彼此邻近的瓦片合并成一个遥感影像瓦片数据集，并能够根据合并服务节点的性能分配合并数据集的数目，同时考虑到合并服务集群的扩展性和所产生的数据迁移问题。

（3）设计了顾及瓦片时空特性的瓦片预取技术和缓存置换策略。为了充分发挥缓存服务器的存储性能，提高遥感影像瓦片的读取性能和访问效率，本文制定

了基于深度优先策略和广度优先策略的预取策略，充分利用用户执行地图操作时的停顿时间，根据用户的访问特性将下一步可能访问的瓦片数据预取到服务器本地缓存，提高用户的响应速度和访问瓦片的效率。此外，设计了基于瓦片时空特征价值的瓦片缓存置换策略，利用遥感影像瓦片的访问特性和时空特性，提高遥感影像瓦片的读取性能。

关键词：海量瓦片；瓦片数据合并；缓存置换策略；Ceph 分布式存储系统

Abstract

In recent years, remote sensing images have become one of the most important geographic data for scientific research, industry applications, and Internet companies. In addition, in order to effectively improve the efficiency of rapid sharing, processing, analysis and display of remote sensing image data, image slicing technology has gradually become one of the important methods of image external service provision. Therefore, remote sensing image tiles have also become the basis for external services provided by various industries Data, how to efficiently organize, store and manage massive remote sensing image tiles is the key to providing high-performance services to remote sensing images. The traditional remote sensing image tile storage system based on database and file system not only performs poorly in terms of storage performance, but also is difficult to expand and maintain, and can no longer satisfy the storage of massive remote sensing image tiles. In the new generation of NoSQL databases and HDFS distributed file systems based on the master-slave architecture, due to the single-point failure of the master node, the master node tends to become the performance bottleneck of the entire storage system. At this time, the distributed storage system Ceph based on the idea of decentralized architecture has become the most massive storage of remote sensing image tiles because it is designed as a system with no single point of failure, high performance, high reliability and high scalability. Good choice. But it has obvious performance defects when storing small files like massive remote sensing image tiles. In view of the above problems, based on the predecessor's small file merging scheme, this paper fully considers the spatial and temporal characteristics of remote sensing image tiles, and designs and implements a multi-level optimized storage method TMOSM for massive remote sensing image tiles. The main research contents of this article are as follows:

(1) Multi-level optimized storage method TMOSM for massive remote sensing image tiles is designed. Aiming at the problem of Ceph's poor performance when storing massive remote sensing image tiles and other small files, this paper designs a

multilevel optimized storage method TMOSM for massive remote sensing image tiles based on remote sensing image tile merging and prefetching cache strategy. The read and write performance of remote sensing image tile storage for Ceph distributed storage system is optimized from the aspects of remote sensing image tile merge storage strategy, global mapping index strategy and prefetch cache strategy. Experimental results show that this method can effectively improve the read and write performance of the Ceph remote sensing image storage system.

(2) Propose a combined storage strategy for remote sensing image tiles based on Z curve and consistent hashing. In order to merge remotely sensed image tiles with high spatial correlation and consider the scalability of the merged service cluster, this paper analyzes the deficiencies and defects of the existing data division methods, and proposes a remote sensing based on Z curve and consistent hashing Image tile merging strategy, which can merge the spatially adjacent tiles into a remote sensing image tile data set, and can allocate the data amount of the merged data set according to the performance of the merge service node, taking into account the merge service cluster's Scalability and data migration issues.

(3) Designed tile prefetching technology and cache replacement strategy that take into account the temporal and spatial characteristics of tiles. In order to give full play to the storage performance of the cache server and improve the reading performance and access efficiency of remote sensing image tiles, this paper has developed a prefetch strategy based on depth-first strategy and breadth-first strategy, making full use of the pause time when users perform map operations, according to The user's access feature prefetches the tile data that may be accessed next to the server's local cache to improve the user's response speed and the efficiency of accessing the tiles. In addition, a tile cache replacement strategy based on the spatiotemporal feature value of the tile is designed, and the access characteristics and spatiotemporal characteristics of the remote sensing image tiles are used to improve the reading performance of the remote sensing image tiles.

Keywords: Massive tiles; tile data merge; cache replacement strategy; Ceph distributed storage system

目录

图索引

表索引

1 绪论

1.1 研究背景与意义

随着信息技术的飞速发展，遥感影像逐渐成为科学研究、行业应用以及互联网企业最重要的地理数据之一，以此为基础的地理信息服务被广泛应用到资源环境监测、土地利用规划、农林牧业生产、地质勘测、灾害突发事件预防与应对、军情侦测等社会生活的各个方面（孙家炳，2003）。同时，传感器以及卫星成像技术也在不断提高，单幅影像的空间分辨率、光谱分辨率、辐射分辨率以及时间分辨率变得越来越高，从而使得单幅影像的数据量变得越来越大，因此覆盖同一区域的影像数据量将会更加的大，影像的数据量正在以几何级的速度增长，例如一幅覆盖全国区域范围的 GF2 卫星影像，其数据量就达到 300G 左右，如果是分辨率更高的无人机航拍影像，其数据量将会更大。其次，随着传感器种类的增多，遥感影像的类型变得越来越丰富，从太空卫星到陆地资源卫星再到移动测量设备都可以作为影像的获取来源（原发杰，2013）。近年来，随着网络地理信息系统 WebGIS（Web Geographic Information System）（Yingwei L et.al，2001；Niu et al，2013）以及移动技术的快速发展，如何实现遥感数据的快速共享、处理、分析与显示（张可力，2014），对于遥感影像数据的网络传输以及服务器性能提出了更高的要求。

影像切片是一种有效提高影像数据快速共享、处理、分析与显示效率的技术，通过将遥感影像按照一定的规则重采样或者切割生成许多不同层级的小文件（称之为遥感影像瓦片或者栅格瓦片），建立瓦片金字塔模型，再通过数据库或文件系统的方式对瓦片数据进行管理和存储，以提供给不同的客户端共享（胡庆武等，2009；宋欣等，2012）。单个瓦片文件一般不超过 10M，因此，其在网络传输、数据处理以及共享发布速度等方面与原始遥感影像相比有着巨大的优势（原发杰，2013）。目前很多国内外企业都对其存储的遥感影像数据进行瓦片化，并对外提供服务，如 Google 提供了瓦片计算服务（Gorelick N et al，2017）、中科院遥感与数字地球研究所提出了五层十五级的瓦片结构，并进行遥感影像的分发与计算（姜斌，2018）、“天地图”也以瓦片的形式实现数据共享等（吕雪锋，2011）。

但是，一幅遥感影像数据重采样或切割后生成的瓦片数量非常多，一幅几百兆大小的 GF1 号卫星影像生成的瓦片数量有几千条，总大小为几个甚至十几个 G。海量遥感影像生成的瓦片数量可以达到千万条甚至上亿条，总大小甚至可以达到 TB 级甚至 PB 级（张可力，2014）。如何对海量遥感影像瓦片进行高效地组织、存储和管理是遥感影像对外提供高性能服务的关键所在。因此，设计实现一种针对遥感影像瓦片存储的方法具有重要的意义。

在很长的一段时间内，国内外很多遥感影像瓦片存储系统都采用关系型数据库或者文件目录的方式对海量瓦片进行存储和管理，这很好地解决了海量瓦片的存储管理问题。但是，在某些方面也存在着一些不足之处：①传统的关系型数据库中是以 BLOB 类型字段来存储瓦片数据的，无法建立高效的索引，当瓦片数据量很大时，其查询效率会大大降低；当用文件目录来存储影像瓦片时，不易实现瓦片数据的冗余备份，存在数据丢失的风险；只能利用文件系统的某个磁盘卷，文件系统的空间利用率较低；同时由于单个瓦片数据一般比较小，容易产生大量的磁盘碎片，文件系统 I/O 性能较差（商秀玉，2012）。②整个瓦片存储系统的伸缩性较差：当增加或者删除一个数据库存储节点或者文件目录存储节点时需要迁移大量的瓦片数据，且迁移的周期非常长，同时在瓦片数据迁移的过程中整个系统需要对外停止服务，无法实现数据的在线迁移，存储系统节点的变更对用户是有感知的，非常影响用户的体验。

基于关系型数据库和文件目录存储瓦片数据的不足，目前很多互联网公司以及行业应用部门开始使用分布式文件系统如 Google 的 GFS（Ghemawat S et al，2003）、HDFS（Hadoop Distributed File System）（Shvachko K et al，2010）、淘宝的 TFS（2013，赵洋）等以及新一代 NoSQL 数据库，如 HBase、BigTable（Chang F，2008）、MongoDB 等来存储和管理海量瓦片。在海量瓦片数据的查询、备份以及整个系统的伸缩性方面表现优异。此外，它们还具有较好的扩展性。但是这种存储方式仍然有一些不足之处：①目前大多数分布式文件系统都是基于大文件存储来设计的，由于单个遥感影像瓦片比较小，使用分布式文件系统存储海量瓦片并不能显著提升瓦片的读写性能，仍然会造成较多的磁盘碎片，磁盘空间利用率不高，网络通信开销巨大，同时还会造成海量小文件问题（Li X et al，2011）。②大多数分布式存储系统采用主从架构模式，主服务器一般用来存储文件的元数

据信息，由于需要组织管理海量的遥感影像瓦片，因此，主服务器存储的元数据信息将会非常庞大，容易成为整个存储系统的瓶颈（张毕涛，2016）。

基于无中心化的分布式文件系统构建海量遥感影像瓦片存储系统能够有效地解决主服务器内存大小受限的问题，同时可以使得整个瓦片存储系统没有单点故障，具有高可用性以及较好的扩展性，但是由于瓦片存储引起的海量小文件问题并不能让整个存储系统性能获得较大提升。因此，为了实现海量遥感影像瓦片的分布式存储，同时兼顾整个存储系统的高可用性、高性能以及较好的扩展性，本文主要分析了无中心化分布式文件系统 Ceph 的系统架构、文件读写流程以及 CRUSH 算法的原理等关键技术，并基于这些技术构建了海量遥感影像瓦片的分布式存储系统，同时基于遥感影像瓦片的时空特性设计了一种基于瓦片合并存储和预取缓存的海量遥感影像瓦片多级优化存储方法 TMOSM，实现了一套面向 Ceph 分布式存储系统并且具有高可用性、高性能以及较好扩展性的遥感影像瓦片存储管理系统。

1.2 国内外研究现状

1.2.1 分布式文件系统研究现状

分布式文件系统是存储管理海量遥感影像瓦片的有效工具，也是海量瓦片对外提供服务的基础。许多互联网公司或者开源社区都在不断设计性能更好且更稳定的分布式文件系统，如 HDFS（Shvachko K et al，2010）、Ceph（Ceph Document，2014）、GlusterFS（Gluster Docs et al，2016）、Lustre（Wang F et al，2009）、GFS（Ghemawat S et al，2003）、Haystack（Beaver D et al，2010）、TFS（2013，赵洋）等。但是这些分布式文件系统通常在存储大文件时具有更好的性能提升，在存储遥感影像瓦片这样的小文件时具有明显的缺陷。目前最具代表性的分布式存储系统有 HDFS 和 Ceph。

（1）HDFS

HDFS（Hadoop Distributed File System）是 Google File System（GFS）的实现，是 Hadoop 生态项目的核心子项目，是整个 Hadoop 生态数据存储管理的基础，它具有较好的扩展性、可靠性、维护性以及伸缩性，为海量数据的存储提供

了基础，为超大数据集（Large DataSet）的应用处理带来了很多便利。HDFS 采用主从架构设计模式（Shvachko K et al，2010），主节点是 NameNode 节点，负责管理维护 HDFS 所有数据的元数据信息以及客户端的读写请求，存在单点故障并且不可以无限制扩展；从节点包括多个 DataNode 节点，主要负责数据的存储与备份，可以无限扩容。但是 HDFS 在存储海量小文件时性能往往表现不佳，甚至会出现 NameNode 节点崩溃（Li X et al，2011）。

（2）Ceph

Ceph 是一个高度统一的分布式文件系统，其设计初衷是提供较好的存储性能、可靠性和可扩展性。随着云计算技术的兴起和普及以及开源云计算管理系统 OpenStack 的完善，Ceph 迅速成为了目前最热门的开源分布式存储系统。Ceph 采用了无中心化的架构设计思想，能够无限扩容。此外，Ceph 具有极高的可靠性、可用性和扩展性。Ceph 的高度统一指的是它可以提供多种存储方式，包括对象存储、块存储以及文件存储。Ceph 的无中心化指的是它没有中心结构，没有理论上限，可以无限的扩展，是其与 HDFS 分布式文件系统最主要的区别，也是相对于 HDFS 集群的最大的优势之一。目前 Ceph 社区吸引了全球大量的存储工程师和研究者的参与，使得 Ceph 受到越来越多大公司的重视（周林，2018）。

1.2.2 海量遥感影像瓦片存储方法研究现状

针对海量遥感影像瓦片数据存储方法的研究，国内外很多企业、行业应用部门以及科学研究工作者都做了大量的研究，也提出了很多优秀的遥感影像瓦片存储管理方法。

国外对于遥感影像瓦片存储方法的研究起步较早，早在 2000 年，世界最大的在线公共地图集之一 TerraServer 采用关系型数据库 SQL Server 和存储区域网 SAN 的混合存储架构对遥感影像瓦片进行统一组织、存储和管理（Slutz T，2000）。TerraServer 采用 UTM 区域划分的方法对遥感影像进行重采样生成一系列固定像素大小（200×200）的瓦片，影像瓦片以二进制的形式存储在 SQL Server 数据库表中的列（BLOB）中，每个影像瓦片记录包括它所在的 UTM 景区域、数据主题、分辨率层级以及在该景中的相对位置坐标 X 和 Y（吕雪锋等，2011）。在存储区域网 SAN 中，通过将磁盘盘阵 SATA 分为三个数据库，每个数据库存储同

一主题的瓦片数据进行冗余备份，从而保证数据的安全性。之后 TerraServer 的存储架构形式有所改变，但是遥感影像瓦片的数据组织形式是不变的（Barclay T，2004）。TerraServer 的物理存储架构如图 1.1 所示。

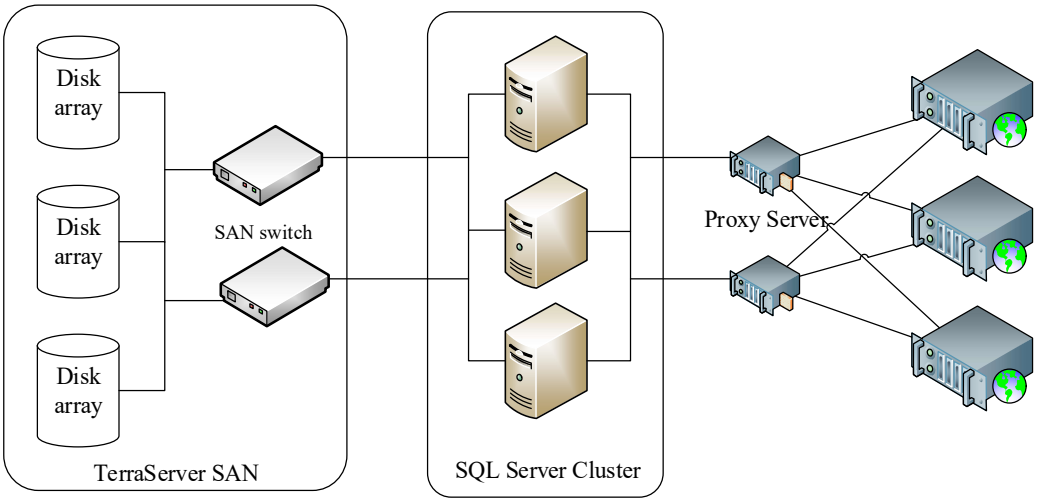

图 1.1 TerraServer 物理存储架构图

2005 年，Google 公司推出了 Google Maps，主要提供矢量地图、卫星影像与 3D 全景影像。Google Maps 采用的遥感影像瓦片存储方法主要依赖于 Google 的云存储体系（Chang F，2008），该体系主要包括 Google 分布式文件系统（GFS）、索引大表（BigTable）、分布式并行计算框架（MapReduce）以及分布式服务器锁服务（Chubby）等。其中 GFS 处于整个体系的最底层，主要用来存储遥感影像瓦片文件（Guo W，2010；Ghemawat S，2003；Dean J ，2008），为了能够快速准确地检索到不同空间范围不同分辨率的瓦片数据，基于索引大表 BigTable 在海量数据中的高效检索机制建立了瓦片数据索引，通过对大表行键的特殊设置可以保证地理上相邻的瓦片在存储上也相邻。此外，大表中的每一行都包含一个列簇，列簇中的每一列表示该瓦片对应的一幅遥感影像。为了提高遥感影像瓦片数据的读写性能，Google 将大表拆分成多个子表部署在 BigTable 集群中，其中主服务器主要用来处理请求的调度与分发，客户端可以直接与子服务器通信进行瓦片数据的读写。Google Maps 的物理存储架构如图 1.2 所示。

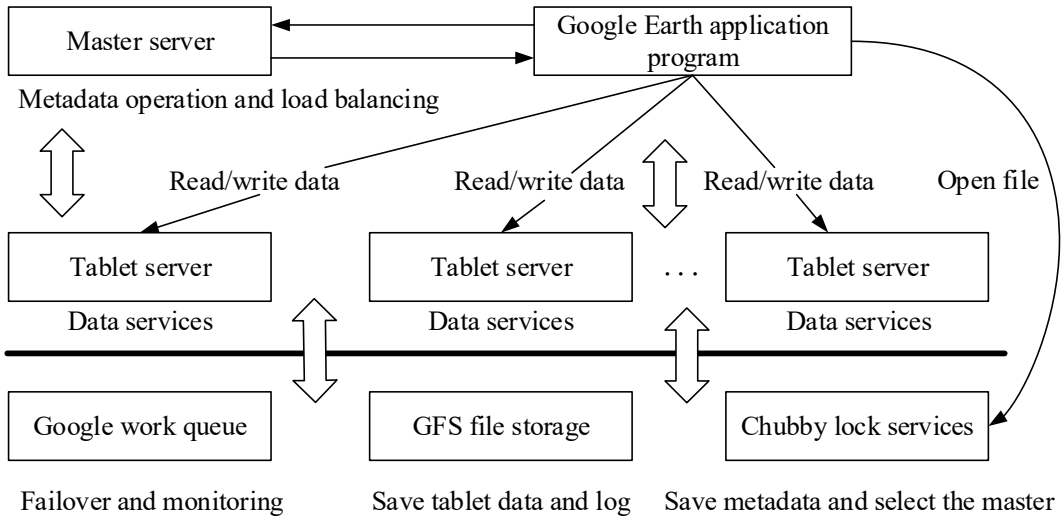

图 1.2 Google Maps 物理存储架构图

之后，NASA 推出并开源了一款三维且可交互的虚拟地球 World Wind，World Wind 采用 NAS 集群文件系统来存储与管理海量遥感影像瓦片（吕雪锋，2011）。这种存储组织模型能够很好地利用遥感影像的空间属性特征，使得地理上相邻的遥感影像瓦片在存储上也相邻。World Wind 遥感影像瓦片存储架构如下图 1.3 所示。当根服务器收到客户端的请求时，首先会通过 HTTP 请求计算出数据集合（DataSet）的类别和位置信息，然后将请求路由到相应的数据处理服务器上进行处理和计算。面对海量用户的并发访问，World Wind 采用了缓存机制，实现高并发情况下系统的快速响应（Bell DG et al，2007）。

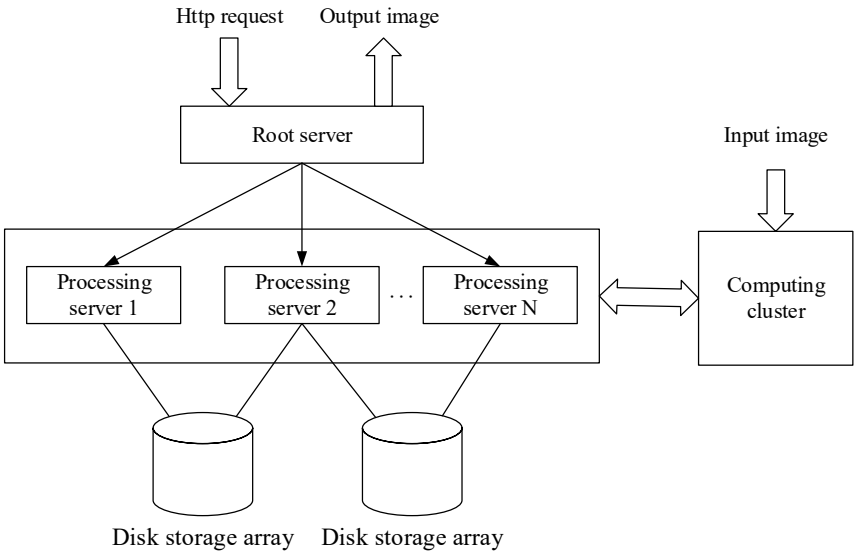

图 1.3 World Wind 物理存储架构图

2013 年，微软发布了一款在线地图服务平台 Bing Maps，该平台在 TerraServer

存储架构的基础上做了改进，采用基于 SQL Server 技术构建的核心数据库 SQL Azure 以及微软云计算平台（Windows Azure Platform）共同存储管理遥感影像瓦片数据（Redkar T et al，2009；Pendleton C et al，2010）。每个瓦片以二进制的形式存储在 SQL Azure 中的 BLOB 中，SQL Azure 主要负责响应用户的数据访问请求并且管理各大组件的自动配置和负载均衡。为了防止遥感影像瓦片数据丢失，微软依托微软云计算平台实现瓦片数据的冗余备份。

与国外发达国家相比，我国在遥感影像瓦片存储研究方面虽然起步较晚，但是在国外研究成果的基础上，并结合我国的实际情况也提出了很多有代表性的遥感影像瓦片存储方法。互联网公司如阿里高德、腾讯地图以及百度地图都提供了相应的瓦片计算及分发模型；行业应用部门如超图公司的 SuperMap IS、武汉吉奥有限公司推出的 GeoSurf 等都有各自的影像切片算法模型，以及“天地图”也以瓦片的形式进行数据共享等；科研机构如中国科学院遥感与数字地球研究所提出了五层十五级的瓦片组织结构。

综合考虑国内外遥感影像瓦片的存储方法可以发现，目前国内外的诸多企业以及行业部门基本上采用传统的 NAS 或者 SAN 技术，采用文件目录层级的的方式对海量瓦片数据进行存储管理。这类方法存储成本高、稳定性好且实现简单，但是在高并发场景下系统读写性能较差，响应时间较长。

除了国内外一些企业以及行业应用部门以外，很多研究学者基于现在的很多分布式文件系统，如 HDFS，以及基于 NoSQL 的非关系型数据库提出了很多遥感影像瓦片存储方法。陈时远（2013）基于分布式文件系统 HDFS 和分布式并行计算框架 MapReduce，提出了采用四叉树快速构建遥感影像金字塔模型的算法，充分利用了数据节点的计算资源。张剑波等（2017）针对传统的栅格数据存储策略不能满足分布式计算环境下粗粒度数据访问需求，应对海量栅格数据计算时效率低下的问题，结合分布式文件系统的存储特点，同时考虑地图代数算子在 MapReduce 阶段以栅格瓦片为单位的计算特点，提出一种基于 Hadoop 分布式文件系统的栅格瓦片存储策略。该方法基本的存储架构如图 1.4 所示，其基本思想是将瓦片作为文件或者将瓦片合并为大文件存储在 HDFS 中，具有扩展性好、数据安全性高、数据管理方便的优点，但是在高并发读写场景下元数据管理容易成为系统瓶颈。

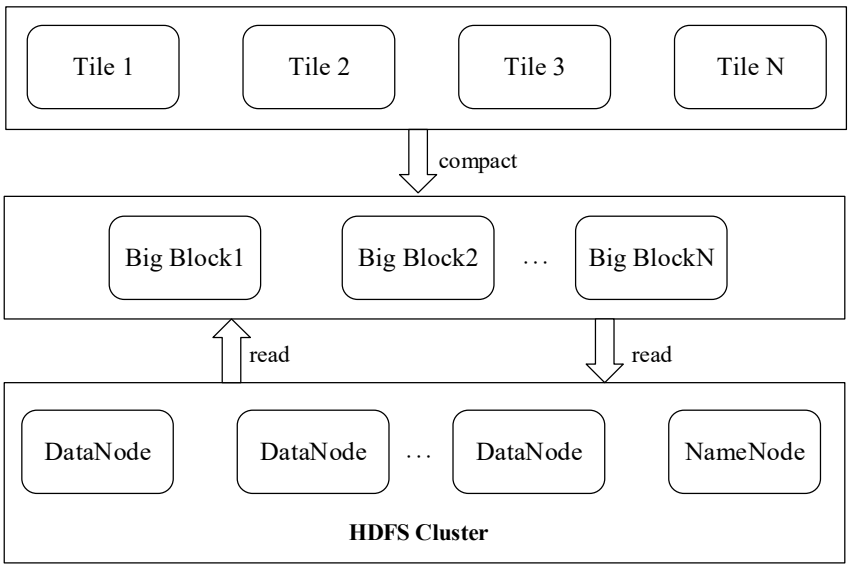

图 1.4 基于 HDFS 的遥感影像瓦片存储方法

李振举等（2015）设计并实现了基于 HBase 的地形数据瓦片存储管理方法，采用四叉树和 Hilbert 编码相结合，对全球地形数据进行切分组织。同时还提出了根据经纬度求地形瓦片行列号以及根据行列号计算地形瓦片 Hilbert 编码的算法。之后张晓兵（2016）也基于 HBase 提出了一种弹性可视化遥感影像瓦片存储系统，可以存储海量遥感影像瓦片数据，同时利用 HBase 的可扩展性并通过 WMS Server 发布标准的地图服务。2018 年，针对海量多源异构遥感数据获取困难的情况，曹梦鸽等（2018）结合森林防火遥感影像数据，设计并实现了基于 HBase 的森林防火遥感瓦片大数据存储方法。孙忠芳等（2015）以 MongoDB 数据库为基础，设计了一套高效实用的瓦片数据库存储方案，大大提高了瓦片数据的读写性能。张飞龙（2016）提出了基于 MongoDB 数据库和分布式文件系统相结合的存储策略对遥感影像瓦片进行存储管理，采用 GeoHash 的编码策略对遥感影像瓦片进行编码，使得地理位置相邻的瓦片在存储上也相邻，结果表明这种存储策略可以提高系统的 I/O 性能以及瓦片的检索效率。马卫春等（2018）为了解决安徽省地理国情普查数据库中遥感影像瓦片访问、加载慢的问题，基于 MongoDB 集群对海量瓦片数据进行统一存储和管理，大大提升了瓦片数据的访问加载性能。这类基于 HBase、MongoDB 等 NoSQL 数据库的遥感影像瓦片存储方法虽然具有快速检索、数据维护方便的优点，但是这些非关系型数据库往往采用主从架构模式，在海量瓦片读写场景下，元数据管理以及主服务器内存容易成为系统瓶颈。

1.2.3 海量小文件存储研究现状

近年来，随着网络通信技术的飞速发展，越来越多的小文件出现在网络中，使得小文件存储的问题越发地突出。因此，许多的研究机构以及互联网企业提出了一系列小文件存储优化方案，其主要的优化方案包括：硬件升级、元数据管理优化、缓存管理技术以及小文件合并存储等（刘爱贵，2013）。

硬件升级指使用存储性能更好的硬件来提高小文件的读写性能，比如，选择具有更高带宽的网络设备来提高存储系统的数据传输效率、使用 SSD 硬盘代替普通的机械硬盘缓解主服务器内存不足的压力等。元数据管理优化主要是针对采用主从架构模式的分布式存储系统，通常采用合并数据读写请求、减少元数据量、提高文件元数据检索效率等多种方法进行优化。缓存管理技术是利用缓存服务器高效存储的特性，将访问较为频繁的小文件缓存起来，从而提高存储系统的读取性能。小文件合并存储技术是目前效果最好且研究最为广泛的小文件优化方法，许多的分布式存储系统如 Facebook 研发的 Haystack、淘宝设计并实现的 TFS 等都采用了这种优化方案。小文件合并存储指的是将具有高度相关性的小文件合并成一个大文件进行存储，从而实现减少文件数量的目标，同时大幅度减少文件的 I/O 次数以及网络通信开销，从而达到小文件高效读写的目的。

Hadoop 为了解决其分布式文件系统 HDFS 在存储海量小文件时性能不佳的问题提出了多种优化方案，如 Hadoop 官方提供的 Hadoop Archive、Sequence files 等合并方案，这些方案虽然能够减少 NameNode 节点的负载压力，但是在索引设计方面存在缺陷，小文件访问性能较低。此外，Korat 等（2012）针对上述小文件优化方案的不足，提出了 CombineFileInputFormat 小文件优化方案，有效改善了小文件读取性能不佳的问题。Liu 等（2015）针对 HDFS 小文件优化方案的缺陷，提出 NameNode 节点缓存部分小文件的思路、Jayakar 等（2014）通过扩展 HDFS 的功能设计并实现了 EHDFS 分布式存储系统，该系统通过在 Hadoop 客户端或 DataNode 节点对小文件合并，并建立小文件到合并文件的映射索引来提高小文件的读写性能。但是这类小文件合并方案在访问小文件时通常需要访问多层索引，其访问延迟较高，读写性能往往不太理想。

针对小文件访问效率低下的问题，赵跃龙等（2012）针对存储空间浪费和小文件访问性能较差的问题，提出了 SFSA 小文件合并方案，该合并方案通过减少

小文件的随机读次数，并且采用缓存技术等方法来提高小文件的访问性能。此外，将相关的小文件聚合在一起写入存储集群，从而达到减少磁盘碎片数量，提高磁盘空间利用率的目的。周恩强等（2013）为了减少小文件访问开销，采用对象命名关联性预取的方法将小文件提前预取到服务器专用缓存区，有效改善了小文件的访问性能。这些小文件方案虽然减少了索引访问次数，但是并没有考虑小文件之间的相关性，并且对于小文件的随机访问，其性能表现不佳。

赵晓乐等（2012）分析了 MP3 文件所包含的元数据信息，设计了在 HDFS 存储系统中有效存储 MP3 文件的模型，通过将小文件合并到 Sequence File 中，并且设计了一种小文件到合并文件的高效索引机制，从而达到小文件高效读写的目的。Liu 等（2009）针对 HDFS 存储 WebGIS 文件性能不佳的问题，设计并实现了 HDWebGIS 系统，充分利用 WebGIS 小文件的空间特性，将相邻的 WebGIS 文件合并成一个大文件，然后写入 HDFS 系统，有效改善了 WebGIS 文件的读写性能。刘高军等（2013）为了提高小文件的存储性能，定期将小文件合并为 Sequence File 进行存储，并且采用 Redis 缓存技术，将大量小文件缓存在 Redis 中，从而提高小文件的读取效率。

1.2.4 遥感影像瓦片缓存技术研究现状

缓存技术能够通过减少客户端与服务端之间的通信开销，有效提高遥感影像瓦片数据的读取效率，遥感影像瓦片存储系统的缓存机制逐渐成为地理大数据存储领域的研究热点之一。目前遥感影像瓦片缓存置换算法主要分为以下几类：

（1）基于传统置换算法改进后的瓦片缓存置换算法

基于传统置换算法改进的瓦片缓存置换算法通常会采用一种或多种传统置换算法思想，如 LFU 算法、LRU 算法等。王浩等（2009）结合 LUR 算法、LFU 算法以及 FIFO 算法思想，提出了基于瓦片访问平均时间间隔的缓存置换算法；涂振发等（2012）提出了最小空间价值的缓存置换算法，将 LRU 算法、FIFO 算法以及基于瓦片对象大小的缓存置换算法相结合。

（2）基于关键因素的瓦片缓存置换算法

基于关键因素的瓦片缓存置换算法往往会选择一个关键因素，然后设置阈值，置换超出阈值范围的缓存数据。Ren 等（2000）提出了基于瓦片数据语义的缓存

置换算法，将瓦片位置语义作为置换的基本依据，置换出离移动客户机最远的瓦片数据；Tu 等（2001）将瓦片视角中心位置作为置换的基本依据，置换出离视角中心最远的瓦片数据；Kang 等（2001）将邻近瓦片的预取概率作为缓存置换的基本依据，实现了一种协同缓存置换算法，将转移概率最小的瓦片置换出去。

（3）基于特征函数计算的瓦片缓存置换算法

基于特征函数计算的缓存置换策略的基本原理是定义一个特征值计算函数，将瓦片主要参数作为特征函数的输入变量，置换出特征值最小的瓦片。王浩等（2009）提出了瓦片老化程度特征值计算函数，将瓦片访问热度和寿命作为特征函数的变量；卢秉亮等（2013）提出了最小访问代价缓存置换算法，综合考虑了瓦片数据的空间代价、时间代价以及数据获取的花费，将访问代价最小的瓦片置换出去；史孝国等（2015）将瓦片数据的大小和访问频率作为特征函数的输入变量，将特征值最小的瓦片置换出缓存；刘佳星等（2017）提出了基于地理单元热度的瓦片缓存置换算法，建立了地理单元热度程度计算函数，将地理单元热度作为特征函数的输入变量。

1.3 研究内容

本论文主要针对 Ceph 分布式存储系统在存储海量遥感影像瓦片时读写性能不佳的问题，设计了基于遥感影像瓦片合并和预取缓存的海量遥感影像瓦片多级优化存储方法 TMOSM。本文首先在现有小文件合并优化存储方案的基础上，结合遥感影像瓦片的时空特性以及分布式环境的特点，提出了基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略；接着，设计了基于布隆过滤器和 FNI-Tree 的全局映射索引，实现遥感影像瓦片的快速检索和访问；然后，根据遥感影像瓦片的访问特性和时空特征价值，提出了顾及瓦片时空特性的预取技术和缓存置换策略，提高了缓存瓦片的命中率，减少了用户的访问延迟，提高了遥感影像瓦片存储系统的读取性能；最后，基于以上的相关技术方法和研究理论，构建了遥感影像瓦片存储系统原型系统，并对与之相关的关键技术进行了实验验证。本文的研究内容包括：

（1）海量遥感影像瓦片多级优化存储方法设计

针对 Ceph 存储系统在存储海量遥感影像瓦片时遇到的小文件问题，本文在

研究现有小文件问题解决方案特别是小文件合并技术的基础上，提出了海量遥感影像瓦片多级优化存储方法 TMOSM，从遥感影像瓦片合并、全局索引设计以及预取缓存策略等多方面对遥感影像瓦片存储系统进行读写性能优化，为大规模海量遥感影像瓦片存储提供了新的思路。

（2）基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略的研究

本文在现有小文件合并方案的基础上，提出了基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略，该策略不仅考虑了遥感影像瓦片的空间特性，同时也充分考虑了合并服务集群的可扩展性以及数据迁移问题，为海量瓦片数据的高效合并提供了解决方案。

（3）顾及瓦片时空特性的预取技术和缓存置换策略的研究

为了减少遥感影像瓦片的访问延迟，充分发挥缓存服务器的存储性能，提高遥感影像瓦片的读取效率，本文充分利用用户浏览地图的停顿时间，并结合瓦片数据的访问特性和时空特性，将用户下一步可能访问的瓦片数据预取到缓存空间，从而提高存储系统的响应速度；此外，提出了基于瓦片时空特征价值的缓存置换策略，提高了缓存瓦片的命中率，从而改善了遥感影像瓦片存储系统的读取性能。

（4）原型系统实现和性能测试

基于上述关键技术，设计遥感影像瓦片存储系统原型系统，并对本文提出的关键技术进行实验验证，从而证明与之相关的关键技术的有效性和可行性。

1.4 论文组织结构

本论文的组织结构如图 1.5 所示，各章节的内容安排如下：

第一章《绪论》。主要研究了在海量遥感影像瓦片存储需求的背景下，现有的各种遥感影像瓦片存储系统的使用情况及其存在的问题，然后针对很多分布式存储系统在存储海量小文件时产生的性能问题，分析了目前使用的很多小文件存储解决方案，特别是小文件合并技术的研究现状及其不足；最后提出本文的主要研究内容，并介绍本文各章节的具体安排。

第二章《Ceph 分布式存储关键技术分析》。主要研究并分析了 Ceph 分布式存储系统，重点介绍了 Ceph 存储系统的系统架构和重要组件，接着从源码层面详细分析了 Ceph 文件读写流程，最后分析了 Ceph 存储海量小文件时的性能问

题。

第三章《海量遥感影像瓦片多级优化存储方法》。主要针对 Ceph 系统的存储特性以及遥感影像瓦片数据的特点，设计了一种采用遥感影像瓦片合并存储策略、全局映射索引策略以及预取缓存策略的海量遥感影像瓦片多级优化存储方法 TMOSM，详细介绍了基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略和基于布隆过滤器和 FNI-Tree 的全局映射索引策略

第四章《顾及瓦片时空特性的预取技术与缓存置换策略》。主要针对如何将瓦片数据进行高效缓存的问题，根据瓦片对象的访问特性和时空特性，制定了基于深度优先策略和广度优先策略的预取策略和基于瓦片时空特征价值的瓦片缓存置换策略。

第五章《原型系统实现与性能测试》。基于遥感影像瓦片合并存储策略、全局映射索引策略以及预取缓存策略等关键技术，构建了遥感影像瓦片存储系统原型系统，并对关键技术的可行性和有效性进行了验证。

第六章《总结与展望》。总结了本文的研究成果，归纳了本文的研究特色，并展望了下一步的研究方向和重点。

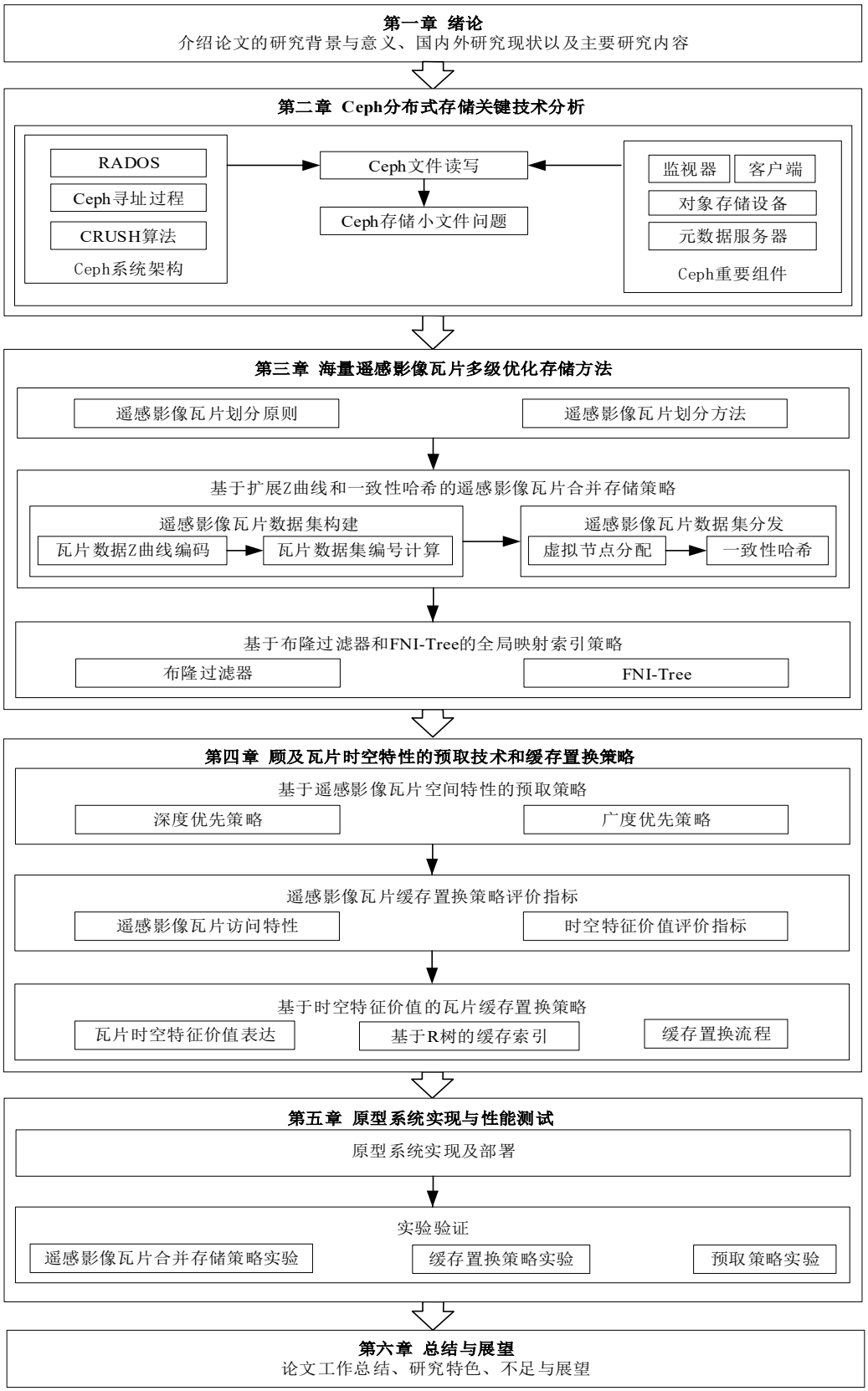

图 1.5 论文组织结构图

2 Ceph 分布式存储关键技术分析

2.1 Ceph 简介

Ceph 项目起始于 2004 年，是其创始人 Sage Weil 在加州大学 Santa Cruz 分校攻读博士学位期间的研究课题之一。2006 年，Sage Weil 在 OSDI 会议上发表了自己的论文并遵循 LGPL 协议（Lesser GUN Public License）开源了 Ceph 的源代码。从此，人们开始逐渐了解 Ceph 分布式文件系统（Sage A. Weil，2006）。2007 年，Ceph 变得越来越成熟并开始进行孵化。2013 年，在 OpenStack Summit 学术会议上，eNovance、Piston、Intel、Cisco、Mirantis、以及 Dell 等大部分存储硬件厂商决定将 Ceph 作为其后备存储方案，从而使 Ceph 引起了更多开发者和用户的关注。

Ceph 作为当前流行的新一代分布式文件系统，具有良好的可靠性、扩展性以及可用性。Ceph 是一个高度统一的可扩展系统，可以支持 PB 级甚至更高级的存储能力，其扁平化寻址的设计方式使得 Ceph 客户端可以和存储集群中的任意节点进行通信；同时支持文件存储、块存储以及对象存储等多种存储方式。与其它分布式存储系统相比，Ceph 拥有很多优秀的特性。Ceph 采用无中心化的设计思想消除了系统对单一中心节点的依赖。其次，Ceph 良好的自动故障修复能力大大提高了整个系统的可用性，这使得 Ceph 受到了越来越多用户的欢迎，可以满足许多行业应用的需求。例如，国内的很多私有云和公有云厂商就使用 Ceph 作为很多云用户的后端存储；国外如雅虎等公司也使用 Ceph 构建分布式存储系统用于博客以及邮箱的存储系统。

2.2 Ceph 系统架构

Ceph 是一个高度统一的分布式存储系统，与传统的分布式存储系统不同，Ceph 对外提供对象存储、块存储以及兼容 POSIX 标准的文件系统接口。此外，Ceph 具有高度的可靠性，易于管理且免费，拥有组织管理海量数据的能力且易于扩展，允许成千上万的客户端访问 PB 级甚至更高级别数据。一个 Ceph 集群（Ceph Storage Cluster）由大量的 Ceph 节点（Ceph Node）构成。Ceph 集群通过

各节点之间的相互通信来保证系统的动态复制及数据迁移。Ceph 集群的逻辑结构以及组成如图 2.1 所示：

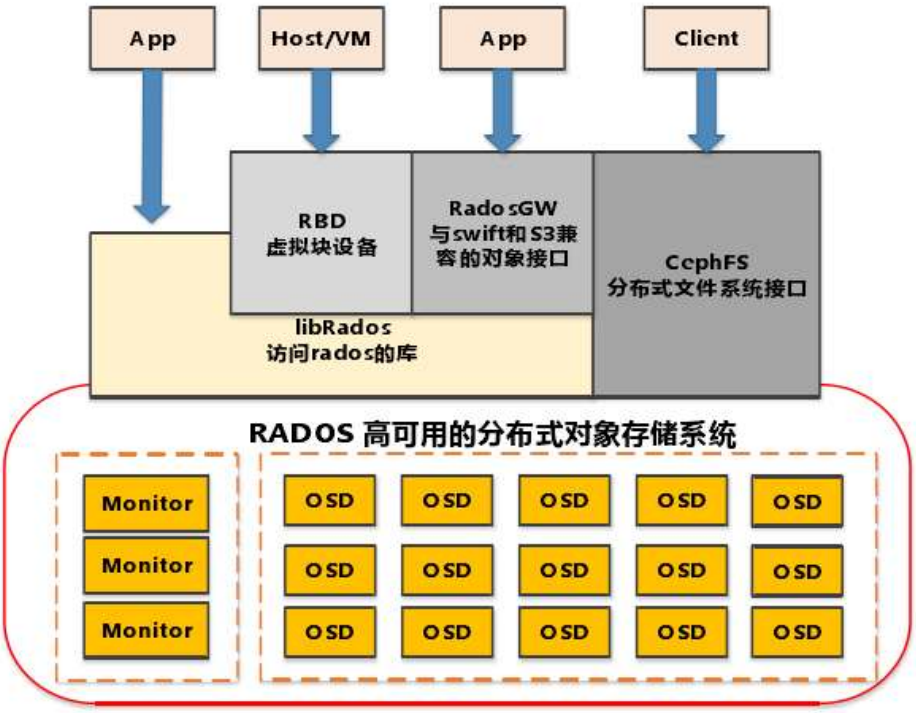

图 2.1 Ceph 集群逻辑结构

RADOS（Reliable Autonomic Distributed Object Store，RADOS）是整个存储系统（也称之为 Ceph 集群）的核心。不管什么类型的数据，RADOS 统一将它们作为对象（object）进行存储，因此，RADOS 可以看作是一个庞大的对象存储系统。RADOS 主要提供 Ceph 集群的高性能、高可靠、高扩展以及高自动化等良好的特性。

librados 是 Ceph 集群提供的基础服务之一，对 RADOS 的原生接口进行了封装并向上层提供应用接口，支持 Java、C、C++、Python、PHP 以及 Ruby 等语言，方便用户访问 Ceph 存储集群和 RADOS，同时也是 RBD、RGW 以及 CephFS 的基础服务。librados API 可以直接操作 RADOS 并且能够使得用户自定义操作 Ceph 集群的接口。

RBD（RADOS Block Device，RBD）是 Ceph 提供的块设备存储接口，与服务器上的普通磁盘一样支持映射、格式化以及挂载等功能。Ceph 块设备还拥有精简配置和快照等特性。

RGW（RADOS gateway）提供了与 Amazon S3（Simple Storage Service）以

及 OpenStack 对象存储 API（Swift）兼容的 RESTful API 接口，同时还提供多租户以及身份权限验证等功能。它允许用户使用不同风格的接口操作 RADOS 数据，大大增强了与其它存储系统的交互能力以及兼容性。

CephFS（Ceph File System）是一个兼容 POSIX 协议的拥有海量数据存储能力的分布式文件系统，使用 CephFS 时至少需要运行一个 MDS。

2.2.1 Ceph 存储系统核心 RADOS

RADOS 是整个 Ceph 存储系统的核心。Ceph 的所有核心功能基本都是由 RADOS 提供的。Ceph 集群支持 RBD、CephFS、RADOSGW 和 librados 等多种数据访问方式，它们都是在 RADOS 层上实现的。此外，RADOS 采用无中心化的架构设计思想，不存在单点故障，具有良好的可靠性、可用性以及自我修复和管理的能力。

RADOS 通常会与 Ceph 客户端进行网络交互，实现数据的读写。比如，Ceph 客户端会通过 CRUSH 算法计算数据的存储位置，然后与 RADOS 中具体的 OSD 节点进行通信，完成数据的写入。RADOS 通常会基于 CRUSH 路由规则将整个文件数据分割成若干个小文件存储在集群中的各个 OSD 存储节点上。如果数据配置了一个以上的副本，RADOS 会通过复制对象，创建副本并将相同对象的副本存储到不同故障域来保证数据的可靠性。此外，RADOS 还可以通过自定义 CRUSH 规则集来保证数据不同的复制集级别。

除了在整个 Ceph 集群内部存储和复制对象外，RADOS 还保证存储数据的状态是一致的。如果相同对象的不同副本的状态是不一致的，RADOS 会通过它提供的自我管理和自我修复机制来进行恢复，此恢复过程对用户是透明的，是无感知的。

2.2.2 Ceph 寻址过程

Ceph 集群的核心功能主要是 RADOS 来实现的，因此，Ceph 的工作原理和若干关键工作流程主要是针对 RADOS 来进行阐述。Ceph 设计了独特的 CRUSH 算法来完成数据的寻址，具体参考 2.2.3 节，能够快速地找到相应的 OSD，有效提高了数据的查询效率。Ceph 系统中的寻址流程如图 2.2 所示，下面先介绍图

2.2 中的几个概念。

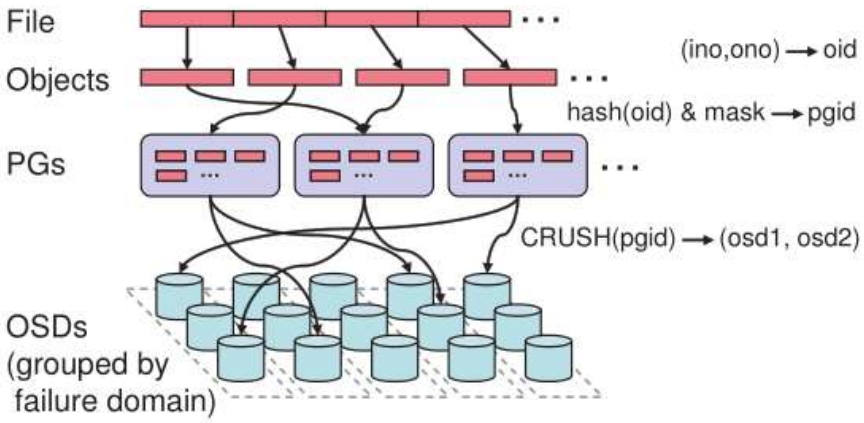

图 2.2 Ceph 系统中寻址流程

File：待写入和读取的数据。

Object：指 RADOS 中的数据存储形式，Object 的最大长度通常是由 RADOS 来限定的，一般为 2MB 或者 4MB，以便实现底层数据存储的组织和管理。当用户向 RADOS 中写入很大的文件时，RADOS 就会按照设置的 Object 的最大长度将该文件切割成许多的数据块，这些数据块是以 Object 的形式存入 RADOS 中。

PG（Placement Group）：主要是对 Object 的存储进行组织和位置映射，是 Object 集合的逻辑组织单元，一个 PG 管理若干个 Object，但一个 Object 只能映射到一个 PG。此外，一个 PG 会被映射到多个 OSD 上，而每个 OSD 上会被承载大量的 PG，PG 和 OSD 之间是多对多的映射关系。通常情况下，一个 OSD 上的 PG 可以达到数百个，一个 PG 最好被映射到 3 个 OSD 上，从而实现数据的冗余备份，保证数据的安全性。

OSD：数据的实际存储单元。

Ceph 的寻址过程主要分为三个步骤：

（1）File->Object 映射：该映射是将用户要操作的大文件分割成 RADOS 能够处理的许多 Object，一方面便于 RADOS 对所存储的 Object 进行高效管理，另一方面对文件切分成许多小的数据块便于 RADOS 对大文件进行并行化处理，例如对文件的读写操作。 分割后的每个 Object 都拥有一个唯一的标识 oid。

（2）Object->PG 映射：当 File 被分割成许多 Object 后就需要将每个 Object 分别独立地映射到一个 PG 中，其映射公式为：

$$hash(oid)\&mask \rightarrow pgid$$

由公式可知，Ceph 集群首先会计算 oid 的哈希值，然后将这个哈希值与 mask 进行按位与计算从而获得最终的 PG 序号。当 Ceph 集群中有大量的 object 和 PG 时，RADOS 可以保证 Object 和 PG 之间的近似均匀映射，同时也能够保证各个 PG 中存储的数据量近似均匀。

（3）PG->OSD 映射：如图 2.2 所示，RADOS 采用 CRUSH 算法将 pgid 映射到若干个 OSD 中，这些 OSD 共同负责存储和管理一个 PG 中的所有 Object。其中 CRUSH 算法是一个伪随机的确定函数，具有非常好的稳定性。当系统中加入新的存储设备 OSD 造成集群规模增大时，大部分的 PG 与 OSD 之间的映射关系不会发生改变，只有少部分 PG 的映射关系会发生变化，从而造成数据迁移。

综上所述，Ceph 集群通过引入 PG 实现了切割后的 Object 与数据存储单元 OSD 的动态映射，从而提高了 Ceph 集群的可靠性，同时也为 Ceph 集群的自动化提供了保证。另一方面也简化了数据的组织管理，降低了系统的管理开销。因此，Ceph 的这种对象寻址机制十分的重要关键。

2.2.3 CRUSH 算法

传统的数据存储机制通常包括数据的存储及其元数据的存储。在传统的存储系统中，每次向存储系统添加新数据时，首先会更新其元数据中该存储数据的实际存储位置，然后再向实际的存储位置写入该数据。研究证明，当存储容量为 GB 到 TB 级别规模时，这种存储机制可以很好地运行。然而当存储规模达到 PB 级甚至 EB 级时，该机制将会出现很多问题。比如可能会由于发生单点故障造成元数据丢失，从而丢失所有存储的数据。目前很多传统的存储系统主要是通过在主节点上保留元数据的多个副本或者复制整个系统的数据及其元数据来提高存储系统的可用性。

在面对 PB 级甚至 EB 级规模的数据存储时，Ceph 使用独特的 CRUSH 算法，它是一种高度扩展的伪随机的数据分布算法，是整个 Ceph 存储系统的核心。与传统的基于元数据或者索引表的存储机制不同，CRUSH 算法能够准确地计算出数据的实际存储位置，从而消除了因管理和存储大量元数据造成主节点服务器出现性能瓶颈的问题。CRUSH 机制仅在数据读写的时候才会进行元数据计算工作，元数据计算也被称为 CRUSH 查找，从而节省了大量的磁盘空间和计算资源。

为了对 Ceph 集群进行数据读写操作，客户端首先会与集群监视器通信获取整个集群映射图。该映射图可以帮助用户了解整个集群的健康状态及其配置信息。数据首先会被转换成具有对象名称和池名称的对象，然后使用 PG 的数量与该对象进行哈希处理获得一个 pgid，最后根据集群映射图、pgid 以及 CRUSH 规则，采用 CRUSH 算法确定要存储和检索数据的一组 OSD 并返回给客户端。之后客户端将直接与该 OSD 通信进行数据的读写操作，所有的计算操作均由客户端来执行，因此并不会影响集群的性能。客户端将数据写入主 OSD 之后，副 OSD 将会从主 OSD 上进行数据的同步，从而实现数据的冗余备份。具体的计算过程如图 2.3 所示：

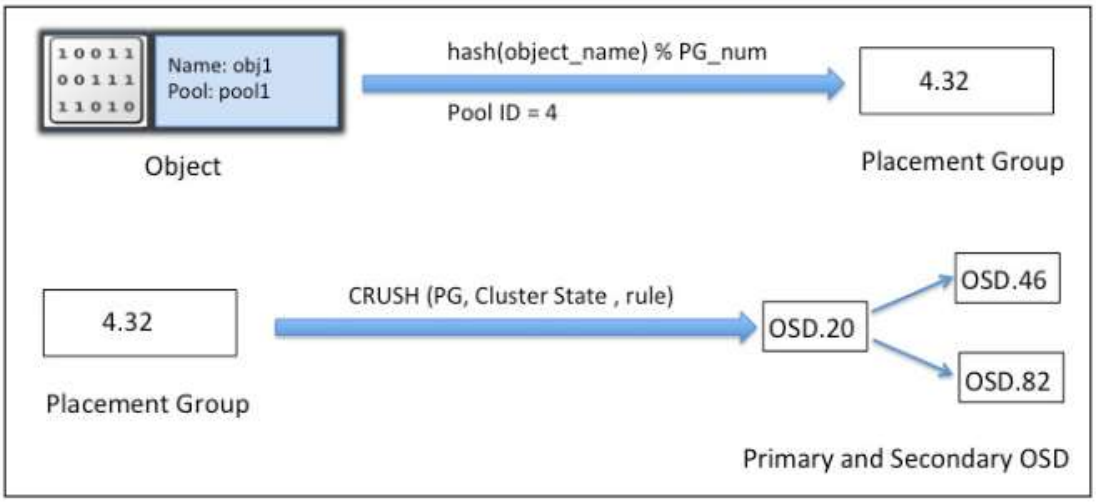

图 2.3 CRUSH 计算过程

2.3 Ceph 文件系统的主要组件

Ceph 分布式文件系统的重要组件（Singh K，2015）包括：客户端（Client），向服务器集群发起请求，遵循 POSIX 标准的文件系统接口；元数据服务器（MDS）集群主要用来管理存储数据的文件目录结构；对象存储设备（OSD）集群用来存储所有的数据，同时还具有副本数据处理以及平衡数据分布等功能；监视器（MON）集群，对整个集群的状态进行监控，维护整个系统的 CrushMap。整个 Ceph 系统的功能是上述这些组件通过相互间网络通信来实现的，具体组成图如图 2.4 所示：

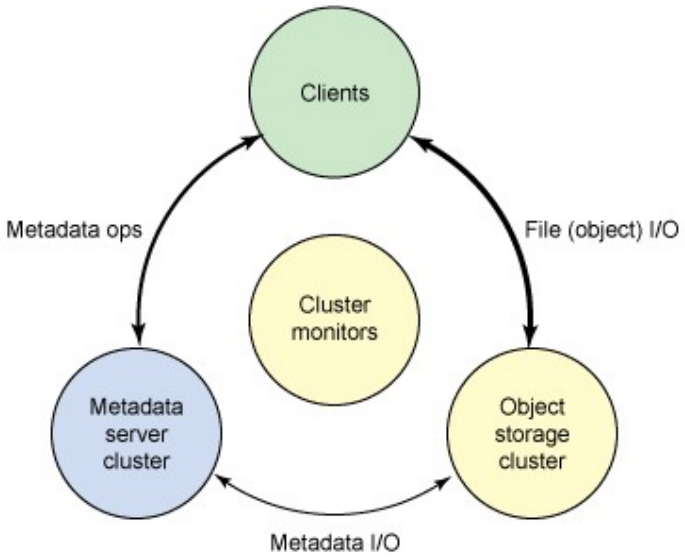

图 2.4 Ceph 主要组件及组件交互图

在图 2.4 中，元数据服务器管理数据存储位置以及在何处存储新数据；客户端会与元数据服务器进行网络通信，确定数据存储位置。元数据存储集群主要负责元数据的存储，客户端和对象存储集群通过网络交互进行实际的文件 I/O。因此，元数据服务器主要负责执行更高层次的文件操作，比如文件的打开和关闭，而对象存储集群主要负责文件的读写。本节将对组成 Ceph 集群的重要组件做详细的介绍。

2.3.1 监视器

监视器通过存储集群配置信息、对等节点的状态以及集群映射图等重要信息来对整个集群健康状况进行监控。其中集群映射图是监视器最为重要的一部分，包括监视器映射图、对象存储设备映射图、放置组映射图、CRUSH 规则映射图以及元数据服务器映射图。Ceph 监视器并不会存储和保留数据给客户端，主要是对集群映射图进行更新。客户端和其它集群节点将会定期检查最新的集群映射图。集群监视器是一个不需要消耗太多系统资源的轻量级进程，大多数情况下，具有大量 CPU、内存和千兆以太网的廉价服务器就足够了。但是监视器节点必须具有足够的磁盘空间来存储集群日志，包括对象存储设备日志、元数据服务器日志以及监视器日志。

2.3.2 对象存储设备

对象存储设备（OSD）是整个 Ceph 集群最重要的组成部分之一，主要负责实际数据的存储，每个 OSD 负责管理一个物理磁盘驱动器，以对象的形式来组织实际数据。Ceph 的主要存储功能是由 OSD 守护程序来完成的。Ceph 集群中包括很多的 OSD，对于任何的数据读写操作，客户端首先会向监视器节点请求整个集群的映射图信息，通过映射图信息就可以获取数据的实际存储位置，之后客户端就可以直接和相应的 OSD 进行 I/O 操作，不会再受到监视器节点的干预，减少了不必要的网络开销，这使得数据的读写过程变得更快。与其它存储解决方案相比，Ceph 的这种数据存储以及检索机制非常的独特。

Ceph 的核心功能，包括数据的可靠性、数据的重新平衡，故障恢复以及数据的一致性都是由 OSD 来完成的。根据配置的复制集大小，Ceph 通过在集群 OSD 节点之间多次复制对象来提高数据的可靠性，使整个 Ceph 集群具有较高的可用性和容错能力。OSD 中的每个对象都有一个主副本和若干个次副本，所有的副本分散存储在处于其它故障域的 OSD 中。如果发生磁盘故障，OSD 守护程序就会自动与其它对等 OSD 节点通信执行数据恢复操作，从其它次副本中根据 PAXOS 算法选出一个次副本作为主副本，同时也会从其它非副本 OSD 中选择一个 OSD 作为次副本，从而维持复制集大小的不变。数据的恢复过程对于客户端是完全透明的，这使得 Ceph 集群更加可靠且一致。

2.3.3 元数据服务器

元数据服务器主要提供元数据服务，Ceph MDS 的功能是由 MDS 守护程序来完成的，它允许客户端可以挂载遵循 POSIX 标准的任何大小的文件系统。MDS 不会向客户端提供任何数据，Ceph 集群的数据服务是由 OSD 来提供的。MDS 主要用来提供 Ceph 分布式文件系统的高速缓存，大大减少了数据的读写操作。MDS 并不存储本地数据，这在某些情况下非常有用。如果 MDS 守护程序崩溃，可以在任何具有集群访问权限的服务器上重新启动 MDS。元数据服务器的守护程序允许配置为主动和被动，主 MDS 服务器通常为激活状态，其它 MDS 为待机状态。如果主 MDS 服务器发生故障，则会从其它 MDS 中选择一个变为激活状态。为了更快地恢复，可以指定一个 MDS 节点作为主 MDS 的备用节点，其内存中

保留着与主 MDS 相同的数据。

2.3.4 Ceph 客户端

Ceph 客户端提供了很多访问接口，包括 RADOS 提供的 librado、基于 librados 封装的块设备接口以及文件系统接口等，是访问 Ceph 存储集群的重要组件。

2.4 Ceph 读写流程

Ceph 系统的 librados 提供了面向对象的高度封装的读写接口，并在此基础上实现了文件的读写流程。本小节基于 Ceph 源码的方式分析了文件的读写流程，主要从 RADOS 客户端的读写对象流程以及 OSD 端读写消息处理流程两部分进行分析。

2.4.1 Ceph 读文件流程

RADOS 客户端的读对象流程首先从 librados.cc 文件提供的读接口 rados_read 开始的，其具体流程如图 2.5 所示。RADOS 客户端提供的读接口 rados_read 收到读数据请求后会使用 librados.cc 文件提供的类 IoCtxImpl 中的 operate_read 方法将读请求封装成一个操作，然后会使用 Objecter.cc 文件提供的类 Objecter 对该操作进行一系列方法处理，最后通过 SimpleMessenger.h 文件提供的 SimpleMessenger 类的 send_message 方法将读操作封装成消息发送到 RADOS 集群。

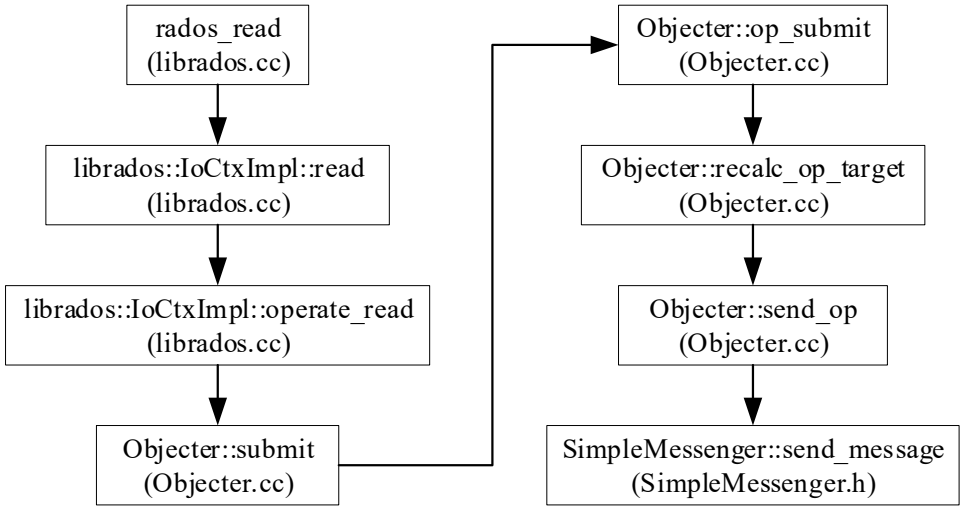

图 2.5 RADOS 客户端读对象流程

OSD 端接收到 RADOS 客户端发送的读数据请求消息后会分为两步对读请求消息进行相应的处理：接收读数据请求消息并把该消息放入优先工作队列；WorkThread 对象实例对工作队列中的任务进行调度执行并返回结果。

OSD 端消息接收模块接收到来自 RADOS 客户端读数据消息请求后会使用 Messenger 类提供的 ms_deliver_dispatch 方法将读请求消息添加到消息分派队列中并对该消息进行分派处理，消息会被分派给 OSD.cc 文件提供的 OSD 类中 dispatch_op 方法进行处理，处理完成后的消息最终会被 OSD 类提供的 enqueue_op 方法将该消息添加到优先工作队列中。具体执行流程如图 2.6 所示。

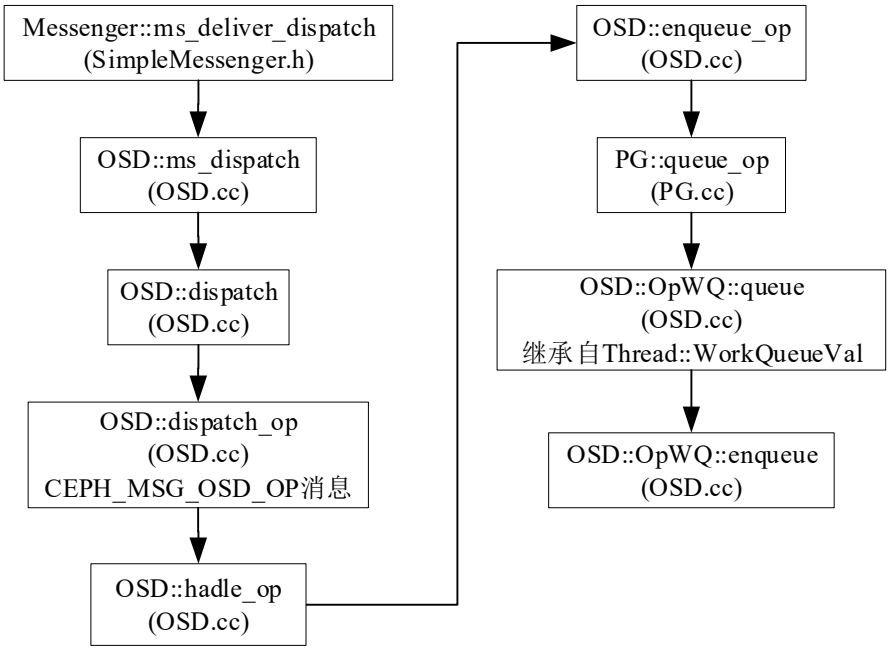

图 2.6 OSD 端读请求分发流程

进入优先工作队列中的消息任务会由 WorkQueue.h 文件中的 WorkThread 实例对象负责进行调度执行并返回结果。优先工作队列中的每个消息任务都会由 ThreadPool 线程池中的某个线程进行处理，每条消息会通过 OSD 类实例对象、ReplicatedPG 类实例对象进行一系列的处理，最后会调用 FileStore.cc 文件提供的 FileStore 类中的 read 方法进行对象数据的读取，并放入 OSD 类提供的 outdata 中，并将其封装成 MOSDopReply 类实例对象，由 SimpleMessenger 类的 send_message 方法返回读取结果。具体执行流程如图 2.7 所示。

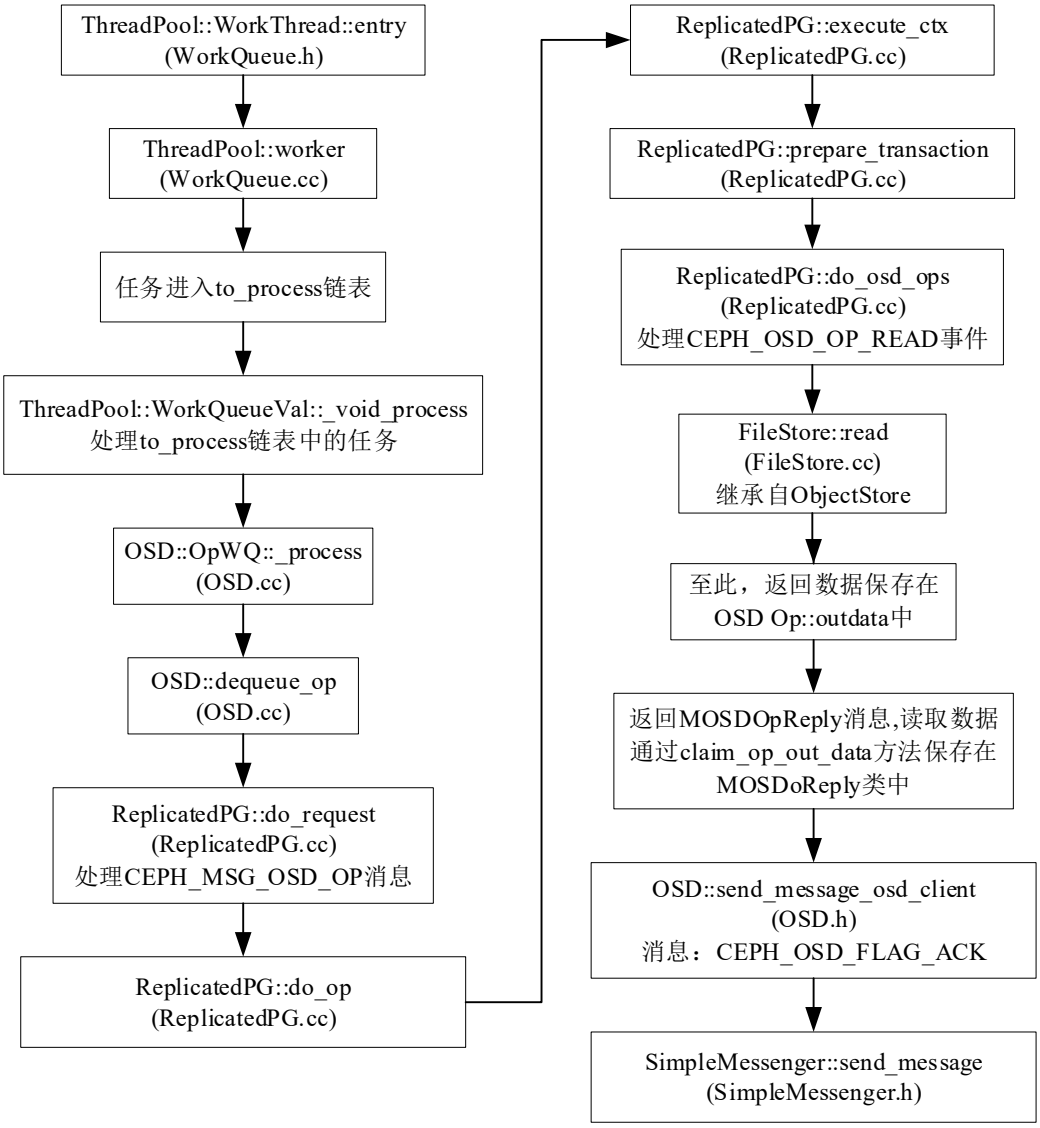

图 2.7 OSD 端读消息处理过程

2.4.2 Ceph 写文件流程

Ceph 写文件流程包括 RADOS 客户端的写对象流程和 OSD 端写消息处理流程。RADOS 客户端的写对象流程是从 librados.cc 文件提供的接口 rados_write_full 开始的，其具体流程如图 2.8 所示。RADOS 客户端提供的写接口 rados_write_full 收到写数据请求后会使用 librados.cc 文件提供的类 IoCtxImpl 中的 wirte_full 方法完成数据写入，然后使用 operate 方法将写请求封装成一个操作，然后会使用 Objecter.cc 文件提供的类 Objecter 对该操作进行一系列方法处理，最后通过

SimpleMessenger.h 文件提供的 SimpleMessenger 类的 send_message 方法将写操作封装成消息发送到 RADOS 集群。

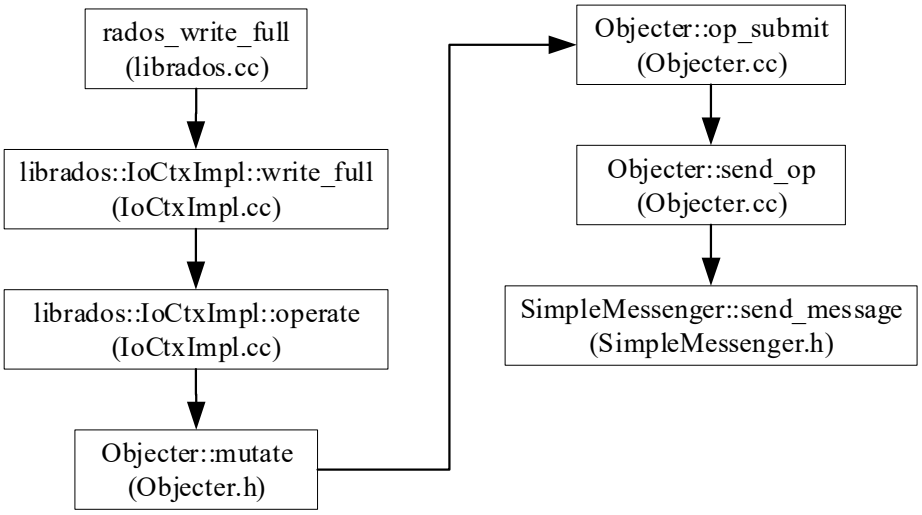

图 2.8 RADOS 客户端写对象流程

OSD 端接收到 RADOS 客户端发送的写数据请求消息后同收到读数据请求的处理过程相似。OSD 端消息接收模块接收到来自 RADOS 客户端写数据消息请求后会将写请求消息添加到消息分派队列中并对该消息进行分派处理。WorkQueue.cc 文件提供的 worker 方法会为每个写消息分配一个线程进行处理，通过 OSD 类实例对象、ReplicatedPG 类实例对象进行一系列的处理后调用 ObjectStore 的子类 Transaction 类中的 write 方法将写请求进行编码并写入缓冲区 bufferlist 中。之后的处理流程如图 2.9 所示，使用 FileStore.cc 文件提供的 FileStore 类中的 do_transactions 方法从 bufferlist 中解析处相应的写操作，然后根据解析出来的数据完成后续多副本对象的写入操作。

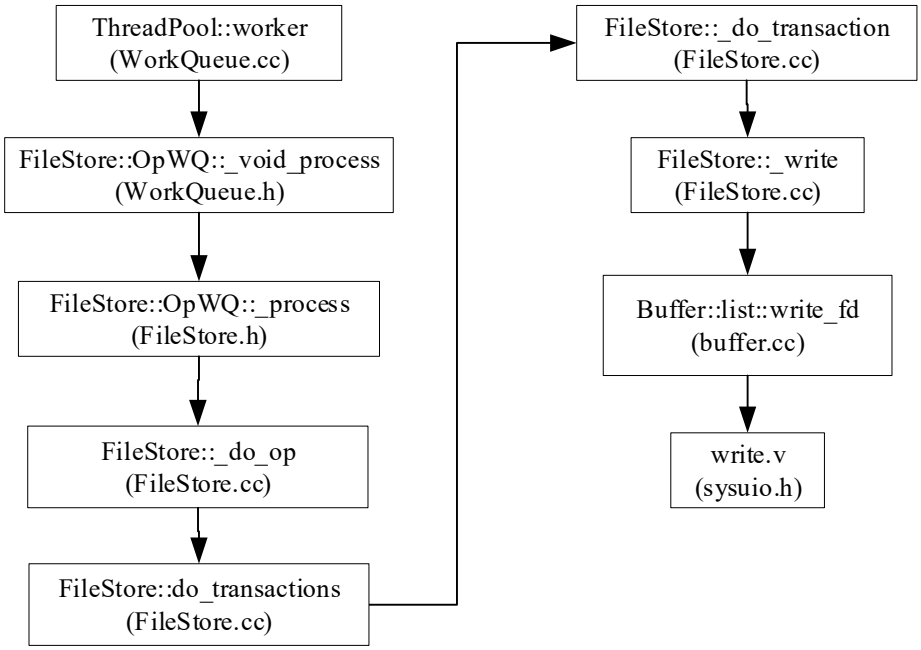

图 2.9 OSD 端写操作处理流程

客户端完成存储对象主副本的写请求后，会把数据同步到该存储对象的其它副本所在的设备中，当所有的副本设备完成了该存储对象的写请求后，再通过主副本设备将写入完成消息反馈给客户端。

2.5 Ceph 存储小文件问题

Ceph 分布式存储系统与传统的分布式存储系统相比，比如 HDFS、MongoDB 以及 HBase 等，并没有采用传统的主从架构设计模式，没有采用单一的元数据服务器来存储数据的元数据信息，而是采用无中心化的设计思想通过元数据集群来管理存储系统的文件结构，同时元数据集群并不会保存数据的元数据信息，因此 Ceph 存储系统并不会出现由于管理大量的元数据信息造成的主服务器性能瓶颈问题。Ceph 系统在存储大文件时会将该文件切分成若干数据块分别存储到不同的 OSD 设备上，通过并行化处理的方式显著提高了 Ceph 系统的读写性能。但是在 OSD 设备上存储对象时为了提高数据的写性能，通常会将数据写请求先记录到日志中，然后再写到 OSD，严重影响了小文件写入性能。另外，Ceph 存储大文件（通常指大于系统默认存储大小的文件）时会将该文件进行切分，切分后的每个数据对象都占用一个存储块；但是当 Ceph 存储小文件时，每个文件对象仍

然占用一个存储块，这就造成了磁盘空间的浪费。因此当存储海量小文件时，Ceph 的磁盘空间浪费问题就会变得越来越明显，同时 Ceph 系统的读写性能也不会得到显著提高。

当存储海量的小文件时，Ceph 的存储性能表现不佳，主要包括两方面的原因：首先是小文件访问效率较低，Ceph 系统访问数据时首先会从元数据集群获取文件存储对象的存储位置，然后向相应的数据存储节点服务器发起数据访问请求，这个过程会造成两次客户端访问 I/O。因此，当面对海量小文件的读写请求时，Ceph 系统会由于大量的访问 I/O 造成整个 Ceph 系统性能下降。其次，Ceph 系统在存储海量小文件时会产生无数的对象，在高并发读写场景下会发生大量的 OSD 寻址计算，浪费大量的 CPU，这也是影响系统存储性能的重要原因。

2.6 本章小结

本章主要介绍了 Ceph 系统的重要组件和系统架构，并详细分析了 Ceph 系统核心 RADOS、Ceph 的寻址过程以及 CRUSH 算法等关键技术。然后详细介绍了构成 Ceph 系统的重要组件，包括监视器、对象存储设备、元数据服务器以及 Ceph 客户端。接着从源码层面详细分析了 Ceph 文件读写流程。最后分析了 Ceph 存储海量小文件时的性能问题，从而为接下来的优化工作打下基础。

3 海量遥感影像瓦片多级优化存储方法

Ceph 存储系统存在的海量小文件问题实质上是由于 Ceph 系统的设计目标决定的，Ceph 系统的设计目标又决定了其设计特性，主要包括：

（1）Ceph 系统是为存储海量的大文件而设计的，能够支持数百 MB 到数百 TB 等超大文件的存储。Ceph 系统的对象设备服务集群可以为超大文件的读取提供足够的带宽。

（2）基于流式的数据访问特性。Ceph 系统主要为海量数据的处理提供存储服务，为应用程序提供流式数据访问，支持应用程序采用批量访问的方式来处理海量数据。

（3）系统故障的快速检测及其自我恢复能力。Ceph 系统能够快速发现集群某个节点的故障并且自动恢复，从而保证 Ceph 系统的可靠性以及高可用性。

目前 Ceph 系统的小文件问题解决方案，主要的优化思路是将小文件合并为大文件存储到 Ceph 分布式文件系统中，主要围绕其上述三个特性进行优化。这种优化方案能够很好地适配 Ceph 对大文件存储友好以及支持流式访问的特性，从而提高 Ceph 系统的读写性能。由于单个遥感影像瓦片较小，通常不超过 10MB，在使用 Ceph 系统来存储管理海量遥感影像瓦片时也会面临着瓦片读写性能不佳的问题。此外，遥感影像瓦片与普通文件不同，除具有普通文件的属性信息外，还具有卫星传感器、经纬度、行列号、层级以及云量等空间属性信息。因此充分利用遥感影像瓦片的特征来合并遥感影像瓦片将是一个具有挑战的问题。

3.1 遥感影像瓦片多级优化存储方法 TMOSM 设计

本章主要提出一种基于瓦片合并与预取缓存的海量瓦片多级优化存储方法（Tile Multilevel Optimal Storage Method，TMOSM）解决 Ceph 系统存储海量遥感影像瓦片时读写性能不佳的问题。TMOSM 主要包含三个优化步骤：（1）使用扩展 Z 曲线对瓦片数据编码，将空间上彼此邻近的瓦片数据合并成一个大文件，然后存入 Ceph 集群。(2)构建基于布隆过滤器与 FNI-Tree 的全局映射索引策略，提高瓦片数据的访问性能。（3）利用瓦片数据的访问特性和独特的时空特性，制定了基于深度优先策略和广度优先策略的预取策略，提出了基于瓦片时空特征价

值的缓存置换策略，减少节点间的网络通信开销，提高瓦片数据读取性能。TMOSM 的瓦片数据读写过程如图 3.1、图 3.2 所示。

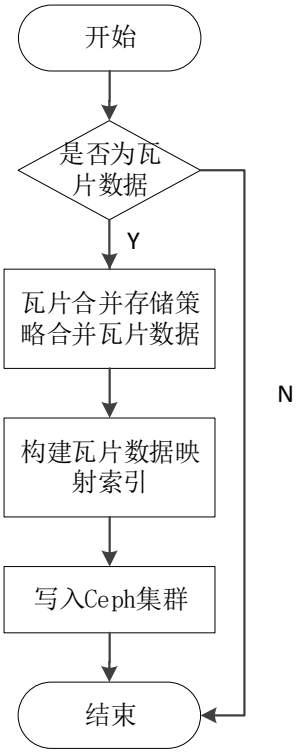

图 3.1 TMOSM 中瓦片数据写入过程示意图

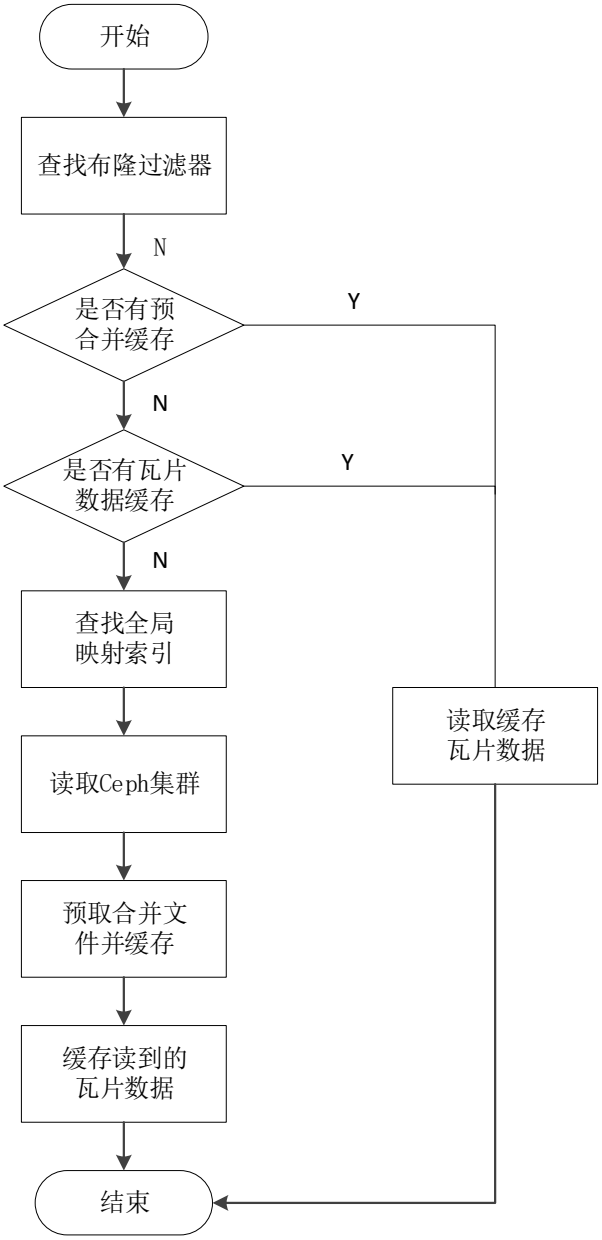

图 3.2 TMOSM 中瓦片数据读取过程示意图

本文设计的海量瓦片多级优化存储方法在原有的 Ceph 系统体系中引入一个逻辑上的客户端代理（Client Proxy），通过对 Ceph 客户端功能扩展来实现 TMOSM 的功能。该客户端代理对外部应用程序透明，能够高效地处理遥感影像瓦片的合并存储策略，同时不影响 Ceph 系统原有的数据访问机制，Ceph 系统的其它客户端、元数据服务集群、监视器集群以及对象设备集群不会感知到该代理的存在，关于代理的相关技术可参考 HAProxy、Nginx 等开源项目。

该方案总体设计结构如图 3.3 所示，Ceph 客户端向 Ceph 集群发起瓦片数据读写请求之前先通过 TMOSM 客户端代理对遥感影像瓦片数据进行一系列的处

理，将多个具有高度时空相关性的瓦片合并成具有特定格式的大文件，然后通过Ceph 客户端写入 Ceph 文件系统。读瓦片数据时先从 TMOSM 代理缓存中查找该瓦片，如果缓存中存在该瓦片则直接返回给应用程序，否则通过全局映射索引找到与该瓦片相关的大文件，然后将构成该大文件的部分遥感影像瓦片预取到TMOSM 代理缓存中，最后返回应用程序待读取的瓦片数据。从而通过 TMOSM 代理提高 Ceph 系统整体读写性能。

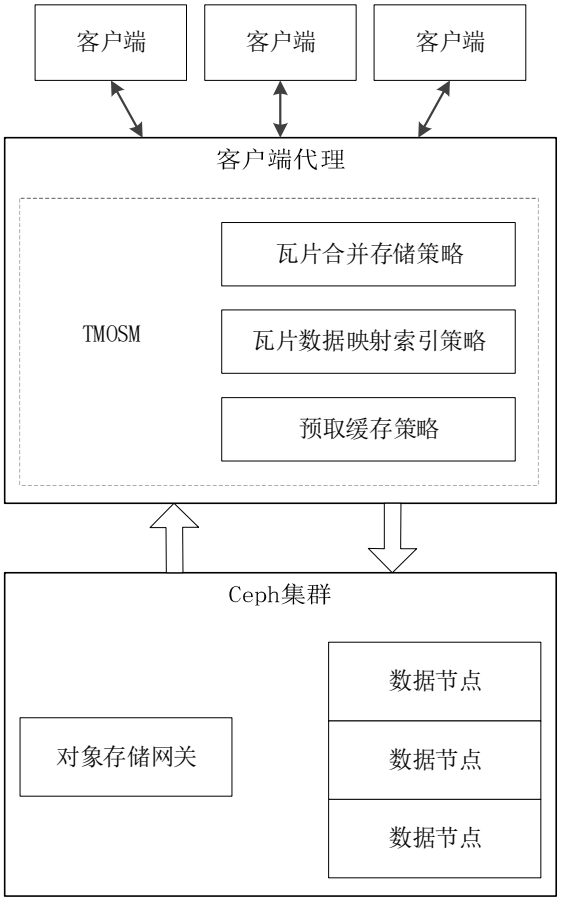

图 3.3 引入 TMOSM 访问体系示意图

3.2 基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略

Ceph分布式文件系统直接存储海量遥感影像瓦片数据时会严重影响Ceph存储系统的整体性能，因此本文采用小文件合并算法对遥感影像瓦片数据进行合并处理。目前存在的许多小文件合并算法虽然能够实现遥感影像瓦片数据的合并，但是并没有考虑瓦片数据彼此之间的时空相关性，通常是将瓦片数据进行随机合

并，造成合并后的遥感影像瓦片读取性能及其缓存时的瓦片命中率都不理想。因此，本文设计的小文件合并算法会充分考虑遥感影像瓦片之间的空间相关性，能够将具有较强空间聚集性的瓦片数据合并成为一个大文件，从而有利于提高遥感影像瓦片数据的读取性能以及瓦片数据缓存命中率。

本文设计的小文件合并算法在执行之前会根据系统配置文件的相关信息启动遥感影像瓦片合并服务集群，每个合并服务计算节点处理具有高度空间相关性的瓦片数据的合并，通常会采用线程池来管理遥感影像瓦片数据的合并过程，每个线程处理一定数目瓦片数据的合并，从而保证合并后的瓦片数据近似相同；其次遥感影像瓦片的合并过程采用多线程异步并发方式，每个线程合并过程互不影响，大幅度提高了遥感影像瓦片的合并效率。该合并算法的核心是遥感影像瓦片数据的划分，是影响 Ceph 分布式存储系统性能的关键技术，目的是将具有高度空间相关性的瓦片数据划分成多个相对独立的瓦片数据集，并且保证数据集内部的瓦片数据具有更好的空间聚集性，最后均匀地分配到执行合并操作的合并服务节点。

3.2.1 遥感影像瓦片数据划分原则

哈希法、轮询法等目前常用的数据划分方法通常是针对一般的非空间数据而言，仅仅考虑了分配给各节点数据数目的平衡，但是在分布式环境下，这些方法往往忽略了各节点的性能差异，从而造成一定程度的数据倾斜。此外，遥感影像瓦片与其它普通文件（比如视频、音频以及图片等）数据不同，遥感影像瓦片数据除具有一般的属性之外，还具有空间特性（谢毅，2011）。因此，采用常用的数据划分方法还会面临以下问题：（1）分布式环境下每个合并服务节点的性能是有差异的，因此在各节点瓦片数目相近的情况下，瓦片数据合并时间可能出现较大差异；（2）由于划分过程中没有考虑遥感影像瓦片的空间特性，每个合并服务节点上的遥感影像瓦片数据空间聚集性较差。综上所述，遥感影像瓦片划分不仅需要考虑瓦片数据的空间特性，尽可能地将空间上彼此相邻的瓦片数据合并到一起，同时也要考虑到瓦片数据合并服务节点的性能和计算的负载均衡。针对以上问题，并结合遥感影像瓦片划分的需求，本文总结了遥感影像瓦片划分的几条原则：

（1）空间聚集性。为提高瓦片数据读取性能，使客户端代理缓存具有较高的缓存命中率，遥感影像瓦片划分需要尽可能地将空间位置相邻的瓦片数据划分到同一数据集中，使得最终合并后的瓦片数据集内部具有良好的空间聚集性。

（2） 负载均衡性。与非空间数据不同，遥感影像瓦片往往具有很强的空间聚集性。因此，遥感影像瓦片数据的划分不仅需要考虑分配到合并服务节点瓦片数据集的数量，还要考虑多核异构条件下各节点的性能和负载情况，从而提高整个系统资源的利用率。

（3）可扩展性。随着遥感影像瓦片数据的快速增长，往往需要对合并服务集群规模进行横向扩展。为保证集群扩展后数据的一致性以及集群负载的再平衡，通常需要进行部分瓦片数据的迁移（何中林，2006）。

3.2.2 遥感影像瓦片数据划分方法

完整的遥感影像瓦片应该包含以下信息：（1）瓦片数据空间信息，即经纬度坐标，这也是瓦片数据区别于其它非空间数据的本质特征，可以通过瓦片数据的行列号计算出来；（2）瓦片数据来源的拍摄日期；（3）瓦片数据分辨率信息，主要通过瓦片数据的层级来体现；（4）瓦片数据来源的卫星类型信息；（5）瓦片数据来源的传感器类型信息；（6）其它辅助数据，如瓦片数据的满覆率、含云量等。综合考虑遥感影像瓦片数据的基本属性以及空间特性，可以设计不同的遥感影像瓦片数据划分方法。遥感影像瓦片划分方法主要包括以下三类：

（1）基于瓦片数据基本属性的一维数据划分

瓦片数据的基本属性主要包括瓦片数据来源的卫星、传感器、拍摄日期、分辨率等非空间属性。该类划分方法能够保证每个合并服务节点分配到的瓦片数目基本相同，但是没有考虑瓦片数据的空间特性，合并后的瓦片数据集内部空间聚集性较差，严重影响瓦片数据的读取性能。该类划分方法主要包括轮询法、范围划分法以及哈希算法等，。

① 轮询法

轮询法是将瓦片数据轮流分配到每个合并服务节点。因此，对于相关性较差的数据集而言，该方法可以保证分配到每个合并服务节点的瓦片数据对象总数基本相同，并且具有良好的合并效率和数据读取性能。

② 范围划分法

范围划分法是对瓦片数据某个属性字段值的范围进行划分，当属性字段值均匀分布时，该方法能够取得不错的划分效果。

③ 哈希划分算法

哈希划分算法包括简单哈希和一致性哈希。简单哈希划分算法通常是指通过划分字段哈希值与合并服务节点数取模的方式建立划分字段哈希值与合并服务节点的映射关系，从而确定瓦片数据应该被分配的合并服务节点。该方法虽然简单高效，但是其扩展性差，当合并服务集群规模扩大时，需要迁移大部分瓦片数据（董献伦，2016）。

一致性哈希划分算法（Consistent Hashing）是由 David Karger 等于 1997 年首次提出，其最初的设计目标是为了解决因特网中的热点（Hot Spot）问题。该算法对简单哈希划分算法进行了改进，当合并服务集群规模扩大时，只需要迁移哈希环上某几个节点的部分瓦片数据，大大减少数据的迁移量。但是，当合并服务节点较少时又容易造成瓦片数据分配不均匀（杨彧剑，2011），通常会在一致性哈希算法中引入虚拟节点的概念，每个虚拟节点对应一个合并服务节点，而一个合并服务节点往往会根据自身性能对应多个虚拟节点。通过在一致性哈希算法中引入虚拟节点的方式可以实现在合并服务节点较少时遥感影像瓦片合并的负载均衡。

（2）基于行列号的瓦片数据划分

瓦片数据的行列号信息表示了瓦片数据的空间属性，其行号表示瓦片的纬度信息，列号表示瓦片的经度信息。基于行列号的瓦片数据划分方法主要包括行列号相加法以及行列号异或法（原发杰，2013），目的是将具有相同行列号的瓦片数据分配到同一个合并服务节点，这类划分方法简单且高效，但是没有考虑到瓦片数据在地理分布上的聚集性差异，往往导致单个合并服务节点计算负载过大，从而大大降低遥感影像瓦片数据的合并效率。

（3）基于空间填充曲线的瓦片数据划分

空间填充曲线是由数学家 Peano 在 1890 年首次提出，其本质上是一种空间降维方法（程昌秀，2012），该曲线能够保证空间上相邻的遥感影像瓦片在填充曲线上也是相邻的，其基本思想是用一条连续的曲线依次穿过每个空间单元，并

且保证每个空间单元只穿过一次。最具代表性的填充曲线包括 Z 曲线和 Hilbert 曲线，采用 Hilbert 曲线划分后的遥感影像瓦片数据集内部虽然具有较好的空间聚集性，空间特性损失较少，但是瓦片划分的时间复杂度较高，在遥感影像瓦片合并过程中会消耗更多的 CPU 计算，影响瓦片写入性能（喻凯，2017）。与 Hilbert 曲线相比，Z 曲线虽然会损失瓦片数据较多的空间特性，但是在保证瓦片数据合并性能的前提下仍然具有不错的划分法效果。两种曲线的生成示意图如图 3.4 所示。

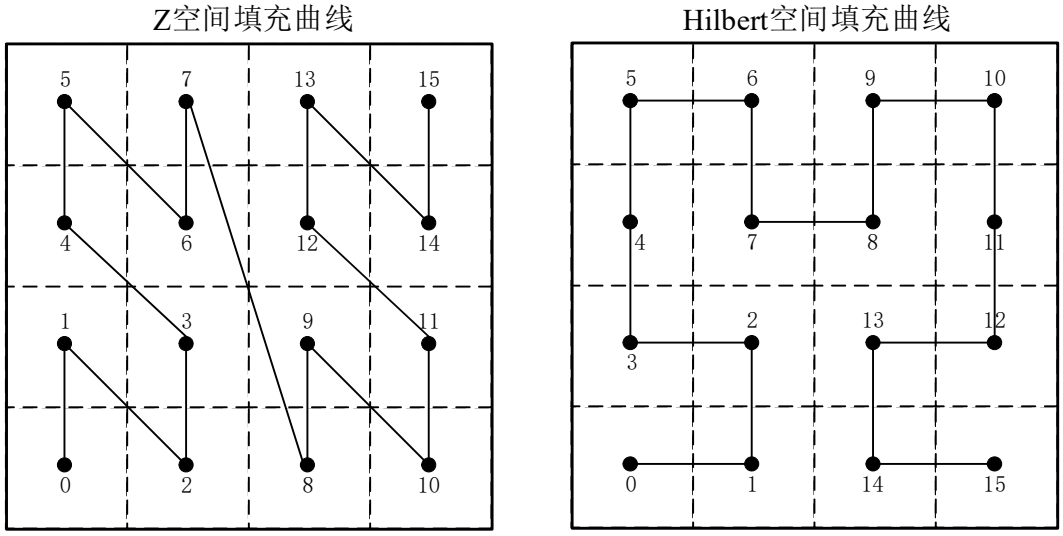

图 3.4 基于 Z 填充曲线和 Hilbert 填充曲线的空间编码

3.2.3 基于扩展 Z 曲线和一致性哈希的遥感影像瓦片数据合并策略

综合目前存在的各种遥感影像瓦片数据划分方法可知，基于 Z 曲线的瓦片数据划分方法能够充分利用瓦片数据的空间特性，将空间上相邻的瓦片划分到相同数据集。另一方面，Z 曲线编码简单高效，在保证较好划分效果的同时能够显著提高瓦片数据的合并效率。但是，图 3.4 中关于 Z 曲线编码的计算方法仅仅考虑了瓦片数据的行列号信息，忽略了瓦片数据的缩放层级，因此可能会产生大量重复的编码值。本文对传统二维 Z 曲线编码方式进行扩展，采用了一种扩展二维 Z 曲线编码的方法。而一致性哈希划分算法在集群负载均衡性以及可扩展性方面表现良好。因此，本文提出了基于扩展 Z 曲线和一致性哈希算法的瓦片数据划分策略，将两种数据划分方法的优劣性进行互补，在取得不错划分效果的同时又可

以兼顾合并服务集群的负载均衡和扩展性。该策略主要分为两部分，分别是瓦片数据集的构建和瓦片数据集的分发。

3.2.4 基于扩展 Z 曲线的遥感影像瓦片数据集的构建过程

（1）基于扩展 Z 空间填充曲线对瓦片数据进行编码

传统二维 Z 空间填充曲线能够将多维空间数据集映射到一维空间并且保持良好的空间聚集性，是一种非常重要的空间映射方法（聂云峰，2012）。二维 Z 曲线编码原理如图 3.5 所示，

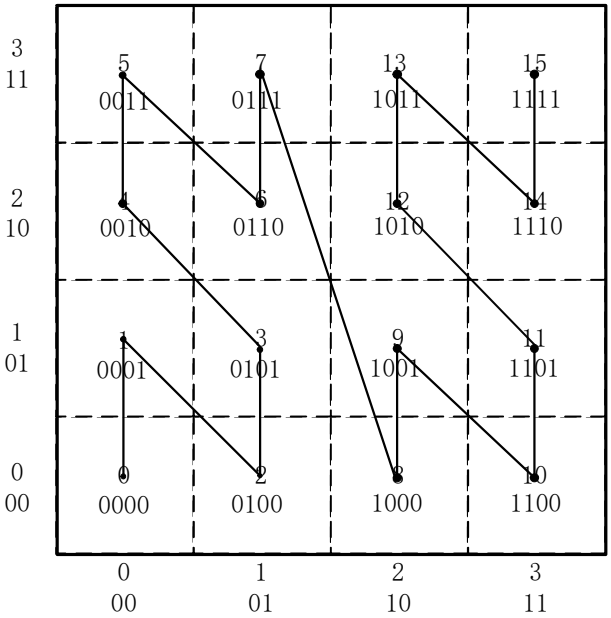

图 3.5 二维 Z 曲线编码原理示意图

若二维空间中的网格单元为：

$$G = \{X = (X_{n-1},\cdots,X_1X_0)_2, Y = (Y_{n-1},\cdots,Y_1Y_0)_2\}$$

则该网格单元在曲线上的位置序号称为该网格的 Z 值：

$$ZValue(G) = (X_{n-1}Y_{n-1}X_1Y_1X_0Y_0)_2$$

将各网格单元的 Z 值按照升序的方式依次相连即是该二维空间的 Z 曲线。

二维 Z 曲线的编码效率较高，Z 曲线的编码值与瓦片数据的行列号之间的映射关系比较简单，但是这种编码方式忽略了瓦片数据的缩放层级，因此可能会产生大量重复的编码值。由于瓦片金字塔模型是一种典型的 2.5 维可变分辨率层次模型，如果直接运用 3 维 Z 曲线对其进行编码，往往容易造成最终的瓦片 Z 编码值不连续并且跳跃较大（聂云峰，2012）。针对上述问题，本文采用一种扩展

二维 Z 曲线编码方法，瓦片数据的 Z 编码值采用 64 位的长整型来进行存储，其比特分配如图 3.6 所示：

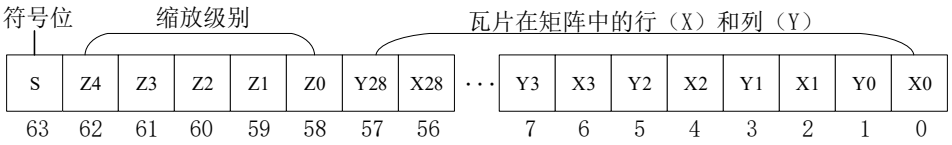

图 3.6 Z 曲线编码值各比特位分配示意图

其中：最高位表示符号位；次高 5 位表示瓦片数据的缩放层级，可表示的缩放层级范围为 0-31 级，其中有效缩放层级为 0-29 级；低 58 位表示瓦片数据的行列号信息，可表示最大范围为$2^{29} \times 2^{29}$的格网矩阵，最多可以存放2.88×10^{17}张不同的遥感影像瓦片，基本可以满足海量遥感影像瓦片数据的编码计算。该存储方法能够保证相同缩放层级的瓦片数据 Z 曲线编码值是连续的，并且可以有效避免大量瓦片 Z 曲线编码值的重复。扩展二维 Z 曲线的编码计算方法如图 3.7 所示：

```
public static long getZValue(String row, String col, String level) {
    long[] magic = { 0x5555555555555555L,
                     0x3333333333333333L,
                     0x0F0F0F0F0F0F0F0FL,
                     0x00FF00FF00FF00FFL,
                     0x0000FFFF0000FFFFL};
    long zValue;
    long tileX = Integer.valueOf(row);
    long tileY = Integer.valueOf(col);
    long tileZ = Integer.valueOf(level);
    tileX = (tileX | (tileX << 16)) & magic[4];
    //填充行坐标
    tileX = (tileX | (tileX << 8)) & magic[3];
    tileX = (tileX | (tileX << 4)) & magic[2];
    tileX = (tileX | (tileX << 2)) & magic[1];
    tileX = (tileX | (tileX << 1)) & magic[0];
    tileY = (tileY | (tileY << 16)) & magic[4];
    //填充列坐标
    tileY = (tileY | (tileY << 8)) & magic[3];
    tileY = (tileY | (tileY << 4)) & magic[2];
    tileY = (tileY | (tileY << 2)) & magic[1];
    tileY = (tileY | (tileY << 1)) & magic[0];
    tileZ = tileZ << 58;
    //填充层级
    zValue = (tileX | (tileY << 1)) | tileZ;
    return zValue;
}
```

图 3.7 扩展二维 Z 曲线编码计算

（2）瓦片数据集的构建

瓦片数据集是遥感影像瓦片数据划分算法的基本数据单元，是瓦片数据合并算法的基本计算单元，也是 Ceph 瓦片数据存储集群的的基本存储单元，是由 Z 曲线编码值连续的的瓦片数据所构成的数据集，因此基于扩展 Z 曲线编码的瓦片数据集可以保证该数据集内部的瓦片具有良好的空间聚集性。瓦片数据集内部瓦片的数目是影响整个存储集群读取性能的重要因素。如果该数据集内部瓦片数目过多，那么合并后的瓦片数据内部包含更多彼此相邻的瓦片，虽然可以提高缓存系统中瓦片数据的命中率，改善瓦片数据的读取性能，但是往往容易造成缓存热点问题，严重时有可能发生缓存穿透。此外在大量用户高并发访问的情况下，

同一个瓦片数据集被多个客户端同时访问的概率就会大大增加，数据一致性的维护开销成本就会增大。如果数据集内部瓦片数目过少，将无法充分利用 Ceph 存储系统各对象存储设备的磁盘空间，造成大量的磁盘空间浪费，此外，也无法充分发挥集群网络性能，浪费大量的网络带宽。另一方面瓦片数据集太小也不能很好地利用瓦片数据的空间特性，造成存储系统内瓦片数据相对离散。因此，可以通过调整数据集内部瓦片数目的大小，来实现瓦片数据集空间聚集性与分布式存储系统资源利用率之间的平衡。其中遥感影像瓦片对应的瓦片数据集序号$tileSetID$如公式（3.1）所示：

$$tileSetID = \left[\frac{ZValue}{N}\right] \tag{3.1}$$

其中，$ZValue$为瓦片数据的 Z 曲线编码值，N为瓦片数据集内部瓦片的数目。

3.2.5 基于一致性哈希算法的遥感影像瓦片数据集分发过程

完成遥感影像瓦片数据集构建后，本文使用一致性哈希算法对瓦片数据集进行哈希映射，并且在传统一致性哈希的基础上引入了虚拟节点，充分考虑了各合并服务节点的性能差异，从而实现分布式环境下海量遥感影像瓦片数据的高效合并。

（1）虚拟节点的配置策略

虚拟节点的配置策略主要包括虚拟节点数量的确定和虚拟节点的分配。

① 虚拟节点数量的确定

假设合并服务集群包括N台合并服务节点$\{M_1, M_2, M_3, \cdots, M_n\}$，性能分别为$\{P_1, P_2, P_3, \cdots, P_n\}$，$M_1$对应的虚拟节点集合为$S_1$，$M_2$对应的虚拟节点集合为$S_2$，依次类推。由于瓦片数据合并服务节点主要是将相同瓦片数据集内部的瓦片合并成大文件，然后写入 Ceph 存储系统。因此可以采用合并服务节点相应磁盘的 I/O 能力作为该合并服务节点的初始性能指标，其中服务器磁盘的 I/O 能力可以通过磁盘的 IOPS 值来反映。本文定义的合并服务节点负载能力指数如公式（3.2）所示：

$$L_i = \left[\left({}^{P_i}/_{U}\right) \cdot 10^m\right] \tag{3.2}$$

其中，m是计算精度表征指数，该指数越大，表明L_i计算越精确。U是负载能力基数，由用户指定，与P_i的单位相同。此外，由公式（3.2）可知，$L_i \propto P_i$，因此，合并服务节点的L_i越大，其负载性能也越高。由一致性哈希算法的均衡性可得合并服务节点M_i的负载压力$C_i \propto |S_i|$，因此可配置$|S_i| = L_i$。该配置策略与直接配置服务节点的方法相比，能够更好地满足集群性能上的负载均衡。

② 虚拟节点的分配过程

为了保证任意一个虚拟节点都能分配到对应的合并服务节点，并且是唯一的合并服务节点，虚拟节点的分配应当满足如下条件：

$$S_1 \cup S_2 \cup S_3 \cdots \cup S_n = R \quad (3.3)$$

$$S_1 \cap S_2 \cap S_3 \cdots \cap S_n = \emptyset \quad (3.4)$$

为了减少虚拟节点和合并服务节点映射的内存开销，同时降低虚拟节点内存管理的复杂度，虚拟节点和合并服务节点的映射关系如表 3.1 所示：

表 3.1 虚拟节点分配

合并服务节点	虚拟节点
M_1	$0 \le s < \|S_1\|, s \in Z$
M_2	$\max(S_1) \le s < \sum_1^2 \|S_i\|, s \in Z$
•	•
M_n	$\max(S_{n-1}) \le s < \sum_1^n \|S_i\|, s \in Z$

（2）遥感影像瓦片数据集的映射

虚拟节点与合并服务节点之间的映射关系构建之后，即可以通过一致性哈希算法将瓦片数据集依次映射到每个合并服务节点。本文使用每个瓦片对应的瓦片数据集$tileSetID$作为其映射的 key 值。

遥感影像瓦片数据集的映射流程如图 3.8 所示，主要分为下面八个步骤进行：

① 初始化一致性哈希算法计算精度表征指数m以及合并服务集群负载能力计算基数U，然后依次测试各合并服务节点$\{M_1, M_2, M_3, \cdots, M_n\}$的 I/O 性能指标，同时根据公式（3.2）计算各合并服务节点的负载能力指数$\{L_1, L_2, L_3, \cdots, L_n\}$；

② 根据本文采用的虚拟节点配置策略依次配置各合并服务节点对应的虚拟节点数量$|S_i| = L_i$（$i = 1,2,3,\cdots,n$）；

③ 根据表 3.1 构建虚拟节点与合并服务节点间的映射关系；

④ 初始化虚拟节点总数$N = \sum_{i=1}^n |S_i|$；

⑤ 获取遥感影像瓦片数据$ZValue$；

⑥ 获取遥感影像瓦片数据集$tileSetID$；

⑦ 根据瓦片数据集$tileSetID$和虚拟节点总数N，采用一致性哈希算法获得对应的虚拟节点v；

⑧ 根据映射关系查找对应的合并服务节点M_k，将该瓦片分配至节点M_k。

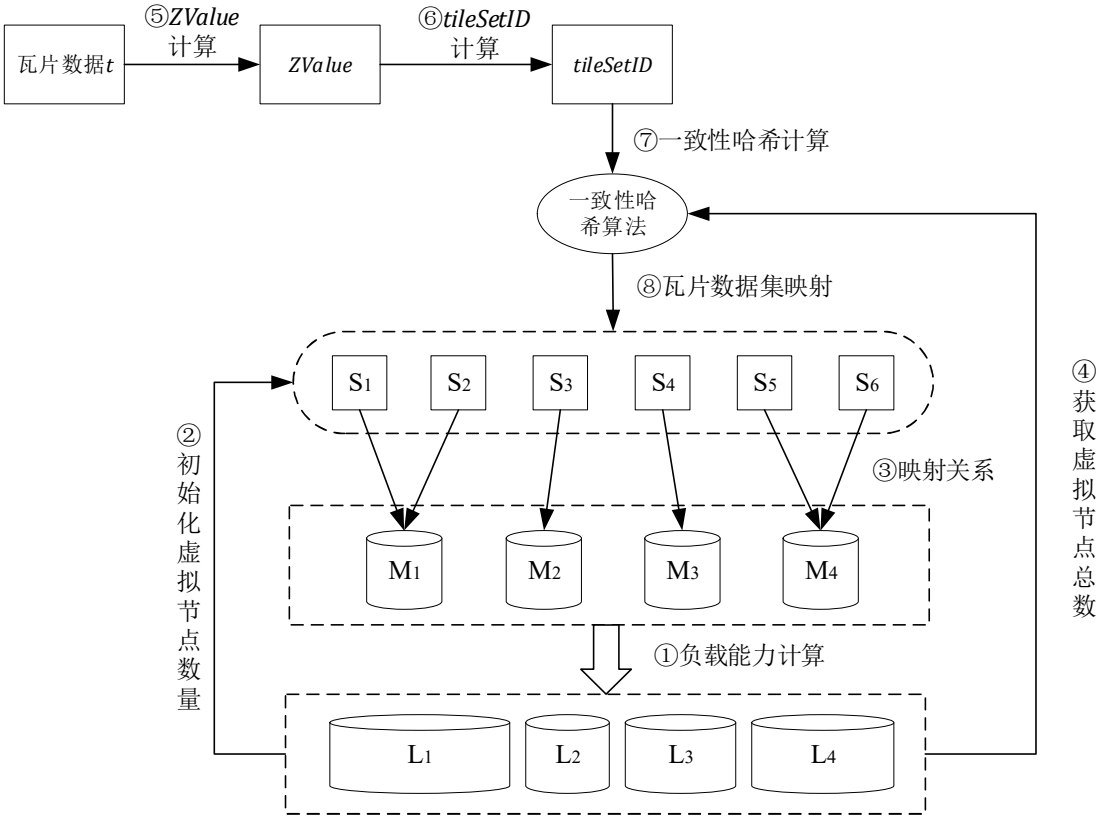

图 3.8 遥感影像瓦片数据集映射流程示意图

3.3 基于 Avro 的遥感影像瓦片合并文件存储结构

对于遥感影像瓦片合并文件的输出格式，目前众多的瓦片数据合并方案中并没有特殊说明，仅通过简单的追加写的方式将瓦片数据不断向目标合并文件中追加，同时记录瓦片数据在合并文件中的起始位置和偏移长度，从而完成合并文件中瓦片数据的随机读写。但是这种随机读写的访问模式往往性能较差，并且在分布式环境下无法保证数据的一致性。

本文对合并文件的输出格式进行了优化，瓦片数据经过合并服务节点合并之后将被处理为 Avro 格式的大文件。Avro 是 Hadoop 生态的一个子项目，也是

Apache 中一个独立的项目，它是一个基于二进制数据传输的高性能中间件，也是一个支持大规模数据存储的数据序列化产品，支持文件的快速压缩以及随机访问，并且可以为其包含的每个瓦片数据提供同步标识，主要目的是在 Avro 文件中快速定位和访问瓦片数据记录（Lu X，2013）。客户端只需要通过合并文件名和目标瓦片数据的同步标识就可以快速访问该小文件，不需要读取整个合并文件再进行顺序查找。Avro 格式的合并文件格式如图 3.9 所示：

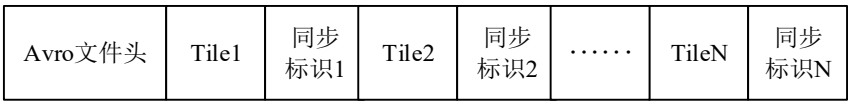

图 3.9 Avro 格式合并文件示意图

其中，Avro 格式文件头用来标识后面每个数据项的格式，即每个数据项都是由“瓦片数据+同步标识”组成的。文件头之后存储的就是真正的瓦片数据，每个瓦片数据都会拥有一个唯一的整数同步标识，该同步标识是由 Avro 中间件自动生成的，并且在合并文件中逐渐递增。此外，Avro 中间件还提供了相应的接口，能够通过该同步标识直接访问合并文件中相应位置的瓦片数据。

遥感影像瓦片数据集是 Ceph 存储系统的基本存储单元，瓦片数据的元数据信息都保存到了瓦片数据集中，其中瓦片数据的元数据包括瓦片名称、瓦片生产日期、瓦片行列号以及瓦片数据内容等，如表 3.2 所示：

表 3.2 瓦片数据元数据字段

字段	类型	说明
tileName	String	瓦片名称
dateTime	Date	瓦片生产日期
tileLevel	int	瓦片缩放级别
col	int	瓦片行号
row	int	瓦片列号
zValue	long	瓦片 Z 曲线编码值
tileSetID	long	瓦片数据集编号
contents	byte	瓦片数据

3.4 基于布隆过滤器和 FNI-Tree 的遥感影像瓦片数据映射索引策略

当若干遥感影像瓦片合并成一个大文件时需要生成瓦片数据到合并文件的映射关系，该映射关系的主要作用是通过要查找的瓦片数据可以快速地查找到它所在的合并文件。为了实现瓦片数据的快速检索和访问，本文采用基于布隆过滤

器（Bloom Filter）（Lee L，1982）和全局映射索引 FNI-Tree 相结合的索引策略。

布隆过滤器的数据结构比较简单，主要用来判断需要查找的瓦片数据是否存在，目前许多高级的程序设计语言都拥有稳定高效的实现。为了减少瓦片数据到合并文件的全局映射索引查找次数，本索引策略引入了布隆过滤器。对于未命中布隆过滤器的瓦片数据访问请求将直接返回用户查找结果，只有命中布隆过滤器的瓦片数据访问请求才会进行全局映射索引查找，从而减小每次瓦片数据访问请求都要访问客户端代理的压力。

为了记录瓦片数据与合并文件的对应关系，且具有较高的瓦片数据查找效率。目前许多的小文件合并方案采用哈希表作为全局映射索引，但是在海量遥感影像瓦片的存储场景中，使用哈希表进行全局索引映射可能会产生如下问题：

（1）不同的瓦片数据名可能产生相同的哈希值，从而发生哈希冲突。

（2）哈希表最好需要提前分配好内存空间，如果瓦片数据哈希值分布不够均匀或者哈希值范围较大，将会引起大量的内存空间浪费。其次，由于遥感影像瓦片数据增长较快，经常需要对哈希表进行动态扩容，通常会移动大量的数据，从而严重影响系统性能。

为了实现瓦片数据到合并文件全局映射索引的快速查找，本文采用了适用于海量遥感影像瓦片快速查找场景的全局映射索引结构 FNI-Tree（File Name Index Tree），该索引结构避免了传统哈希表的缺陷，将使用哈希值进行数组映射改为对素数序列进行取模运算，采用树状结构实现瓦片数据映射索引的动态更新和检索。下面首先说明 FNI-Tree 的理论基础。

定理 1：对于任意n个素数（$n \in N$），它们存在关系$P_1 < P_2 < P_3 < \cdots < P_{n-1} < P_n$，假设存在$m \le s_1 < s_2 < m + \prod_{i=1}^{n} P_i$，$(m, s_1, s_2 \in N)$，那么对于任意的$i \in [1, n]$，$(s_1 \ mod \ P_i) = (s_2 \ mod \ P_i)$不可能总成立。证明如下：

证明：假设定理 1 不成立，即对于任意的$i \in [1, n], (s_1 \ mod \ P_i) = (s_2 \ mod \ P_i)$将总能成立，即$s_1$与$s_2$会存在以下关系：

$$s_1 = l_{1i} \times P_i + a_i, s_2 = l_{2i} \times P_i + a_i, (l_{1i}, l_{2i}, a_i \in N)$$

令：

$$S = s_2 - s_1 = (l_{2i} - l_{1i}) \times P_i = x_i \times P_i > 0, (x_i \in N)$$

可知x_i是正整数并且存在：

$$x_1 \times P_1 = x_2 \times P_2 = \cdots = x_n \times P_n = S$$

则S是一个包含因子$P_1, P_2, \cdots, P_n$的合数，是这些素数的公倍数，即：

$$S = P_1 \times P_2 \times \cdots \times P_{n-1} \times P_n \times Q, (Q \in N)$$

可得：

$$S = Q \times \prod_{i=1}^{n} P_i \geq \prod_{i=1}^{n} P_i$$

又由$m \leq s_1 < s_2 < m + \prod_{i=1}^{n} P_i$，可得$0 < s_2 - s_1 = S < \prod_{i=1}^{n} P_i$，与上式矛盾，因此该假设不成立，即证定理 1 成立。

由定理 1 可知：对于给定的不包含重复元素的整数序列，对n个不同的素数依次进行取模运算可以获得不同的余数结果序列。比如整数序列 3、8、16、20、30，对素数 2、3、5 分别取余，可获得 5 组完全不同的余数结果序列（1，0，0）、（0，2，3）、（0，1，1）、（0，2，0）、（0，0，0），该余数结果序列在 FNI-Tree 结构中分别对应不同的路径。因此能够根据瓦片数据名的哈希值构造 FNI-Tree。具体的 FNI-Tree 结构如图 3.10 所示，该树的特点是每层节点的子节点数目为该层节点对应的素数，该层节点的子节点代表对该层素数的不同取余结果，即对该数据不同的处理路径。例如，第一层根节点对应的素数为 2，则该根节点对应 2 个子节点，对 2 取余结果为 0 的数据将进入左节点处理，对 2 取余结果为 1 的数据将进入右节点处理，依次类推。

本文使用素数序列{2，3，5，7，11，13，17，19，23，29}构造了 10 层的 FNI-Tree，最多可容纳$\sum_{i=1}^{10} \prod_{j=1}^{i} P_j$共 6703028889 个数据。当且仅当两个整数s_1、s_2满足$|s_1 - s_2| = Q \times \prod_{i=1}^{10} P_i, (Q \in N)$时才会发生哈希冲突，而$s_1$与$n \times s_1, (n \in N, n \times s_1 < \prod_{i=1}^{10} P_i)$不会发生哈希冲突；其次，FNI-Tree 可以通过增加素数序列的长度，即直接增加一层节点的方式解决哈希冲突，且实现简单方便。

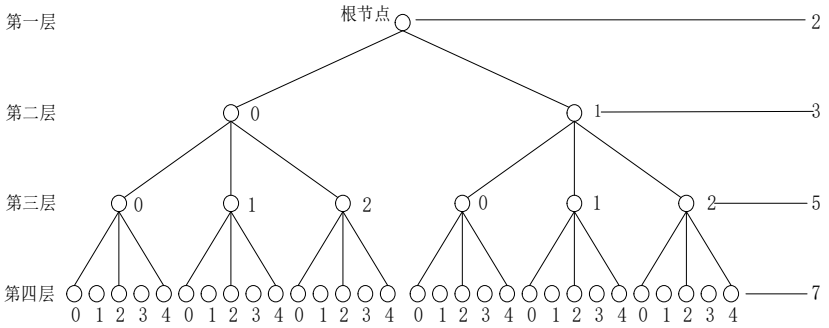

图 3.10 FNI-Tree 结构图

FNI-Tree 中除根节点以外，每个节点都包含瓦片数据名的哈希值、索引数据以及使用标识符等信息。由于遥感影像瓦片数据规模庞大，本文使用运算性能更好、碰撞率较低的 MurmurHash 哈希算法（Fumito Yamaguchi，2013），索引数据包括瓦片名称、合并文件名以及同步标识。其节点结构图如图 3.11 所示：

哈希值	瓦片名称，合并文件名，同步标识	使用标识	子节点引用

图 3.11 FNI-Tree 节点结构示意图

FNI-Tree 只需要初始化根节点，之后按插入规则向树中动态插入节点即可。其插入、删除以及查找的时间复杂度均为 O(n)。此外，FNI-Tree 可以通过增加素数序列长度实现快速扩容，并且树中的节点能够根据使用标识进行复用，并不需要提前分配较大的内存空间，因此，FNI-Tree 索引结构可以很好地适用于海量遥感影像瓦片存储场景。

3.5 本章小结

本章主要针对 Ceph 系统的存储特性以及遥感影像瓦片数据的特点，设计了一种采用遥感影像瓦片合并存储策略、全局映射索引策略以及预取缓存策略的海量遥感影像瓦片多级优化存储方法 TMOSM。遥感影像瓦片合并存储策略充分考虑了瓦片数据的空间特性以及合并服务集群的扩展性，采用扩展 Z 曲线编码和一致性哈希算法的瓦片数据划分策略，高效地将空间上彼此相邻的瓦片数据进行合并，然后将合并后的瓦片数据集存入 Ceph 集群。此外，考虑到遥感影像瓦片的快速检索，设计并实现了基于布隆过滤器和 FNI-Tree 的全局映射索引，对遥感影像瓦片的读取性能进行了优化。

4 顾及瓦片时空特性的预取技术与缓存置换策略

目前越来越多的互联网应用采用数据预取与缓存策略来提高存储系统中数据的读取性能。本文采用遥感影像瓦片数据合并策略，将空间上彼此邻近的瓦片数据合并成大文件，在读取瓦片数据时，会增加通过全局映射索引查找该瓦片数据对应的合并文件以及在合并文件中查找该瓦片数据的过程，相较于直接存储瓦片数据会花费更长的时间。因此，为了提高瓦片数据的访问效率，本文将充分利用用户在进行地图操作时的停顿时间，将用户下一步有可能访问到的瓦片数据预取到服务器缓存，从而实现瓦片数据的预取和缓存。此外，因为缓存区空间通常是有限的，所以不能将 Ceph 系统中存储的所有瓦片数据全部缓存到文件缓存区，必须要充分结合遥感影像瓦片数据的访问特性以及独特的时空特性制定合适的缓存置换策略，保证缓存区中缓存的瓦片数据具有较高的缓存命中率。下面具体论述了瓦片数据预取和缓存的可行性，主要包括：

（1）可预取性

通常用户在查看遥感影像时往往存在短暂的停顿，而且用户接下来要访问的瓦片与当前显示内容具有较强的空间相关性。因此，可以充分利用相邻瓦片之间较强的空间相关性以及用户查看遥感影像的停顿时间，对用户下一步可能被请求到的瓦片数据预取到服务器本地缓存，从而有效提高瓦片数据的读取性能。

（2）可缓存性

通常用户在查看遥感影像时，请求的瓦片数据往往具有一定的时空相关性，此外，由于单张遥感影像瓦片较小、更新时间较短，因此可以将用户访问较频繁的瓦片数据缓存起来，从而提高瓦片数据的缓存命中率，改善遥感影像瓦片存储系统的读取性能。

遥感影像瓦片预取和缓存策略的优劣将直接影响瓦片存储系统的性能。因此，本文以提高 Ceph 存储系统的读写性能为目的，充分考虑瓦片数据的时空特性，设计并实现高效的瓦片数据预取和缓存方案，从而提高 Ceph 遥感影像瓦片存储系统的读取性能。

4.1 基于遥感影像瓦片空间特性的预取策略

瓦片数据预取的基本思想是：采用某种预取策略将用户在未来可能要访问到的瓦片数据提前预取到服务器本地缓存中。这样当用户对已经预取的瓦片数据发起访问请求时，由于该瓦片数据在本地缓存中已经存在，因此可以在请求的第一时间直接获取到，从而大大减少了瓦片数据读取时间，有效地提高了存储系统的读取性能。

4.1.1 遥感影像瓦片加载服务原理

常规遥感影像瓦片加载服务基本原理是，首先需要根据浏览器屏幕的经纬度范围计算出需要加载的瓦片数据的行列号。假设当前缩放层级为N，该层级瓦片分割间隔度数为D。则待加载瓦片数据的行列号计算过程如下：

（1）根据用户请求，获得屏幕左下角和右上角的屏幕坐标，并根据对应的坐标转换公式得到左下角的地理坐标(X_{LB}, Y_{LB})和右上角的地理坐标 (X_{RT}, Y_{RT})。

（2）获得屏幕起始瓦片，即屏幕左下角瓦片的行列号：

$$\begin{cases} col_{LB} = \left\lfloor \frac{X_{LB}+180^{\circ}}{D} \right\rfloor \\ row_{LB} = \left\lfloor \frac{Y_{LB}+90^{\circ}}{D} \right\rfloor \end{cases} \tag{4.1}$$

（3）获的屏幕终止瓦片，即计算右上角瓦片的行列号：

$$\begin{cases} col_{RT} = \left\lfloor \frac{X_{RT}+180^{\circ}}{D} \right\rfloor \\ row_{RT} = \left\lfloor \frac{Y_{RT}+90^{\circ}}{D} \right\rfloor \end{cases} \tag{4.2}$$

（4）获得待加载的瓦片数据行列号集合：$((col_{LB}, row_{LB}), (col_{RT}, row_{RT}))$

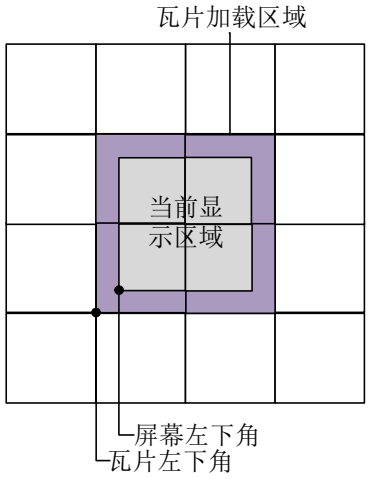

图 4.1 遥感影像瓦片加载区域示意图

4.1.2 遥感影像瓦片预取策略的制定

遥感影像瓦片预取策略主要包括广度优先策略和深度优先策略，其中广度优先策略的目的是预取屏幕显示区域空间相接范围内的瓦片数据，常用于地图平移操作场景下的瓦片数据预取；深度优先策略的目的是预取屏幕显示区域相邻缩放层级的瓦片数据，常用于地图缩放操作下的瓦片数据预取。本文基于广度优先策略和深度优先策略制定了地图平移和缩放操作下的瓦片数据预取策略。

（1）基于广度优先策略的瓦片数据预取策略

为了减少用户在地图平移时的等待时间，提高遥感影像瓦片存储系统的响应速度，需要用户进行地图平移操作时对瓦片数据进行预取。用户在地图平移操作时的预取模型如图 4.2 所示，其中，灰色区域表示当前显示区域范围内的瓦片数据，而白色区域表示用户待预取的瓦片数据。

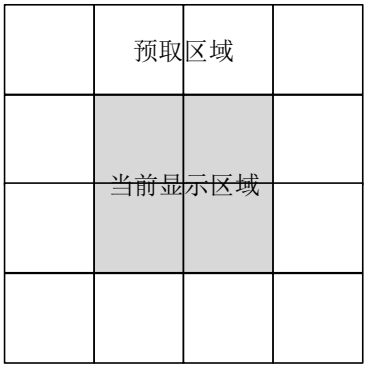

图 4.2 平移操作预取模型示意图

（2）基于深度优先策略的瓦片数据预取策略

为了减少用户在地图缩放时的等待时间，提高遥感影像瓦片存储系统的响应速度，需要用户进行地图缩放操作时对瓦片数据进行预取。用户在地图缩放操作时的预取模型如图 4.3 所示。

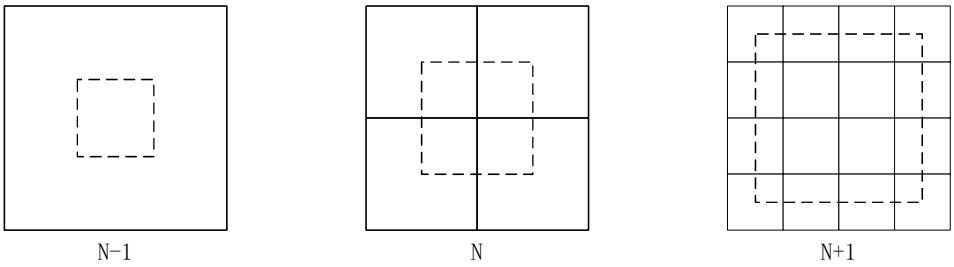

图 4.3 缩放操作预取模型示意图

4.2 遥感影像瓦片缓存置换策略的评价指标

当用户访问某一瓦片数据时，会将该瓦片数据所在的合并文件读取，并将合并文件中部分瓦片数据预取到服务器缓存，这样能够有效提高瓦片数据的读取性能。但是，缓存区空间通常是非常有限的，无法将所有的瓦片数据预取到缓存区。如何制定有效的瓦片数据缓存置换策略，将具有较低访问频率的瓦片数据置换出缓存，保证缓存区中的瓦片数据具有较高的缓存命中率，是目前瓦片数据缓存优化的关键技术。目前国内外的诸多研究机构提出了许多有效的缓存置换策略，其经常使用的瓦片数据缓存置换策略算法如下：

（1）基于瓦片数据访问特性的缓存置换策略

基于瓦片数据访问特性的缓存置换策略最具代表性的算法包括最近最少使用算法（Least Recently Used，LRU）（廖鑫，2008）和最不经常使用算法（Least Frequently Used，LFU）（Gang J Z，2001）。

① 最近最少使用算法（Least Recently Used，LRU）：该算法的基本思想是置换出距离上次访问时间最久的瓦片数据，缓存中最近被访问过的瓦片数据往往会留在缓存，而长时间没有被访问过的瓦片数据容易被置换出缓存（张震波，2006）。

② 最不经常使用算法（Least Frequently Used，LFU）：该算法的基本思想是置换出访问频率较低的瓦片数据，而保留访问热度较高的瓦片。但它有个明显的

缺点是新写入的瓦片数据如果不能频繁地被访问，则无法进入缓存空间。

（2）基于关键因素的缓存置换策略

基于关键因素的缓存置换策略往往会选择一个关键因素，然后设置阈值，置换超出阈值范围的缓存数据。该类算法最具代表性的是基于对象大小的缓存置换策略，其主要思路是优先置换出较大的瓦片数据，从而使得缓存区有较大的剩余空间容纳更多的瓦片对象，越大的瓦片数据越难留在缓存区中，越小的瓦片数据则往往容易长时间停留在缓存区，这种置换算法在缓存空间不够大时容易达到较好的置换效果。

（3）基于特征函数计算的缓存置换策略

基于特征函数计算的缓存置换策略的主要思想是首先定义一个特征值计算函数，当有新的瓦片数据进入缓存区时则会生成一个特征值，并且更新其余相关的瓦片数据特征值。如果缓存区剩余空间不足，将会置换出特征值最小的瓦片数据。该类算法通常将瓦片数据的访问次数、访问时间、数据大小以及空间特性等一种或多种属性作为特征函数的输入变量，能够根据不同的遥感影像瓦片应用场景设计不同的特征函数，是目前最具代表性的缓存置换策略思想。比较有代表性的缓存置换策略包括：

① 基于瓦片寿命和访问热度的瓦片数据置换策略 TCLEPR（王浩，2009），通过缓存中瓦片数据的寿命和访问热度来计算瓦片数据的老化程度，将老化程度较高的瓦片数据置换出缓存。

② 基于地理单元热度的瓦片数据置换策略 GUH（刘佳星，2017），通过缓存中瓦片数据的缩放层级以及所在的地理单元来计算瓦片数据的热度值，将热度值较低的瓦片数据置换出缓存。

③ 基于累计单元时间访问次数的瓦片数据置换策略 Stat（Li R，2015），通过瓦片数据的访问时间间隔和访问次数来累计计算瓦片数据的 stat 值，将 stat 值较低的瓦片数据置换出缓存。

④ 基于最小价值的瓦片数据置换策略 GDLVF（涂振发，2012），将瓦片数据的访问频率、访问时间、数据大小以及空间位置特性作为瓦片价值计算函数的输入变量，将价值最低的瓦片置换出缓存。

4.2.1 遥感影像瓦片数据访问特性

遥感影像瓦片数据的访问特性对瓦片数据的缓存策略有重要的影响，反映了用户对瓦片数据的访问习惯和浏览深度，对缓存数据的一致性和替换策略的高效性有着重要意义。遥感影像瓦片主要包括以下三个访问特性：

（1）局部性特征：用户从服务器接收到的瓦片对象都趋于聚集在一个较小的连续区域中，局部性特征通常分为空间局部性和时间局部性。

① 空间局部性是指在相同的缩放层级内，如果某个瓦片对象被访问过，那么在空间上与其相邻的瓦片在不久的将来会有较大的概率被访问。

② 时间局部性是指如果某个瓦片对象刚被访问过，那么在接下来的一段时间拥有更大的概率被再次访问，且时间越短，被再次访问的概率则越高。

（2）Zipf-like 幂律性：用户在地形漫游时对遥感影像瓦片的访问是不均匀的，瓦片数据的访问频率值与其访问排名之间的关系表现出幂函数关系（王浩，2010）。

（3）主题倾向性（褚信，2016）：用户在较短的一段时间内会对某种主题的瓦片数据感兴趣，并且会产生明显的倾向性。因此，这段时间内该主题的瓦片数据具有较大的概率被访问。

4.2.2 遥感影像瓦片时空特征价值评价指标

影响缓存置换策略的价值指标通常有很多，如瓦片数据的访问频率、瓦片数据的最近一次访问时间以及瓦片数据进入缓存的顺序，然而并没有考虑瓦片数据的空间位置特性及其访问特性对缓存置换策略的影响。而基于特征函数的缓存置换策略虽然构造了不同的特征函数，如概率、老化程度以及热度等，但是这种特征价值指标仅仅反映了瓦片的长期时空特性，缺乏对于短期时空特性的表现，既没有考虑当前请求的瓦片对下一时刻或者其相邻瓦片访问造成的影响，也没有考虑其对覆盖相同地理区域的其它层级的瓦片的影响，无法充分利用瓦片数据访问的时空特性，从而影响瓦片数据的缓存命中率。

因此，本文在基于特征函数的瓦片数据缓存置换策略的基础上，优化了瓦片数据时空特征的表达方式，设计了基于时空特征价值的瓦片数据缓存置换策略，主要包括两个评价指标：空间特征价值和时间特征价值。

4.2.2.1 空间特征价值

缓存系统中瓦片数据的空间特征价值主要包括两部分：同一缩放层级内周围瓦片数据对请求瓦片空间特征价值的影响和覆盖相同地理区域处于不同层级的瓦片数据对请求瓦片空间特征价值的影响。

（1）相同缩放层级内周围瓦片数据对请求瓦片空间特征价值的影响。

在同一缩放层级内，如果某个瓦片被访问，那么其周围的瓦片下一次被访问的概率服从高斯分布，即与当前请求瓦片的距离越近，那么接下来被访问的概率也就越大。如图 4.4 所示，当瓦片（x,y）被用户访问之后，那么与其相邻的瓦片，如图中的阴影部分的瓦片数据将具有更大的概率被用户访问。假设当前访问的瓦片数据为(x_0, y_0)，与其邻近的瓦片数据为(x, y)，那么当前访问的瓦片数据对相邻瓦片数据的空间特征价值的贡献值服从高斯核函数，其贡献值表达式为：

$$w = e^{-\frac{\sqrt{(x-x_0)^2+(y-y_0)^2}}{2\sigma^2}} \tag{4.3}$$

其中，σ为高斯核函数的控制范围，其值越大，高斯核函数的影响范围也就越大。此时，瓦片数据(x, y)的空间特征价值为：

$$SpatialLocationValue(x, y) = SpatialLocationValue(x, y) + w \tag{4.4}$$

瓦片数据(x_0, y_0)的空间特征价值为：

$$SpatialLocationValue(x_0, y_0) = SpatialLocationValue(x_0, y_0) + 1 \tag{4.5}$$

因此，由公式（4.4）和（4.5）可知，在相同的缩放层级内，瓦片数据的空间特征价值实质上是瓦片数据访问次数和其周围瓦片对其贡献值的累加，既充分考虑了瓦片数据的空间特征，同时也体现了瓦片数据访问的短期局部性，与被访问瓦片越近的瓦片将具有较高的概率被访问。

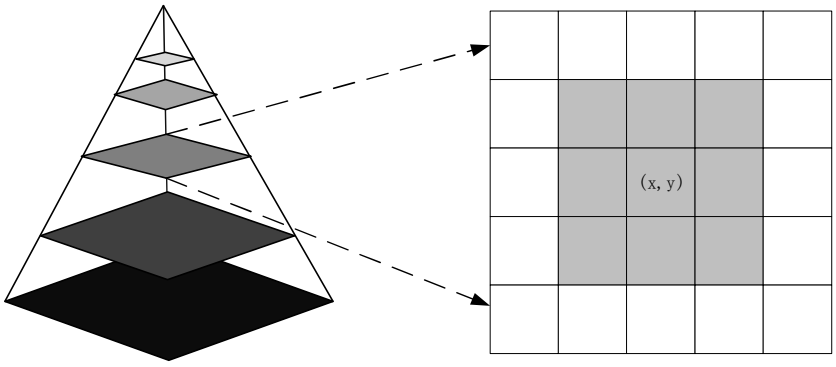

图 4.4 空间特征价值示意图（相同缩放层级）

（2）覆盖相同地理区域不同层级的瓦片数据对请求瓦片空间特征价值的影响。

影像切片技术实现了遥感影像对某一地理区域的多分辨率表达，相同的地理区域可能被缓存区中不同缩放层级的遥感瓦片所覆盖，那么该地理区域受到用户的关注度明显高于其它地区。如图 4.5 所示，S区域被缩放层级为t和$t+1$的不同分辨率的遥感影像瓦片所覆盖，则下一次访问到覆盖S区域的瓦片数据的概率则越高。假设，某种主题的遥感影像瓦片的缩放层级范围为$1\sim M$，当前缩放层级为L，相邻缩放层级为N，如果某个瓦片数据t被命中，那么它对于处于相同地理覆盖范围的第N层的瓦片数据的空间特征价值的贡献值为：

$$w = \frac{1}{2^{2(N-L)}} \tag{4.6}$$

此时第N层的相关瓦片数据的空间特征价值变为：

$$SpatialLevelValue(x,y) = SpatialLevelValue(x,y) + w \tag{4.7}$$

则层级为L的瓦片数据的空间特征价值为：

$$SpatialLevelValue(t) = \frac{1}{2^{2(N-L)}} \times \sum_{y=Y_{min}}^{Y_{max}} \sum_{x=X_{min}}^{X_{max}} SpatialLevelValue(x,y) \tag{4.8}$$

其中，$\frac{1}{2^{2(N-L)}}$表示第N层的瓦片数据相对于层级为L的瓦片权重，(x,y)为N层瓦片数据坐标。

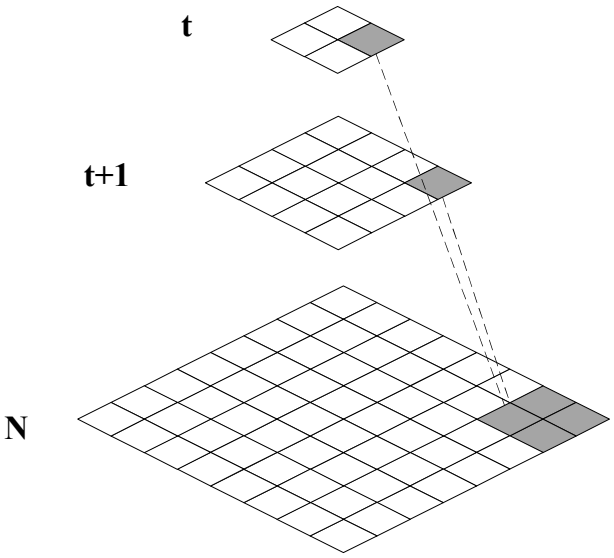

图 4.5 空间特征价值示意图（不同缩放层级）

因此，瓦片数据的空间特征价值最终的数学表达式为：

$$SpatialValue = SpatialLocationValue + SpatialLevelValue \quad (4.9)$$

4.2.2.2 时间特征价值

如果遥感影像瓦片数据上次被访问后，则接下来被再次访问到的概率则越高，该瓦片数据的时间特征价值越大。假设系统当前时间为t_{sys}，瓦片数据最近一次被访问的时间为t_{last}，瓦片数据被访问的总次数为$count_{tol}$，则瓦片数据的时间特征价值表达式为：

$$TimeValue(t) = \frac{count_{tol}}{t_{sys} - t_{last}} \quad (4.10)$$

4.3 基于遥感影像瓦片时空特征价值的缓存置换策略

遥感影像瓦片缓存置换策略的设计是一种多特征价值决策的问题，其本质是在充分考虑了不同特征价值的对缓存对象的影响程度，对不同权重的缓存置换策略方案进行权衡的过程。对各特征价值指标进行加权几何平均是较为常用的信息决策方法。

4.3.1 遥感影像瓦片时空特征价值表达

在进行遥感影像瓦片时空特征价值计算之前，首先需要消除不同量纲对决策结果的影响，本文对空间特征价值和时间特征价值做如下处理：

$$TimeValue(t) = \frac{1}{TimeValue_{max}} \times \frac{1}{t_{sys} - t_{last}} \times count_{tol} \tag{4.11}$$

$$SpatialValue(t) = \frac{1}{SpatialValue_{max}} \times (SpatialLocationValue + SpatialLevelValue) \tag{4.12}$$

其中，$TimeValue_{max}$为缓冲区中瓦片数据时间特征价值的最大值，$SpatialValue_{max}$为缓冲区中瓦片数据空间特征价值的最大值。最后将瓦片数据的空间特征价值和时间特征价值统一量纲后进行比较，计算得到瓦片数据t 最终的缓存价值：

$$CacheValue(t) = \mu_1 \times SpatialValue(t) + \mu_2 \times TimeValue(t) \tag{4.13}$$

其中，μ_1和μ_2分别为缓存中瓦片数据空间特征价值和时间特征价值对应的权重，且$\mu_1 + \mu_2 = 1$。

4.3.2 基于 R 树的遥感影像瓦片缓存索引

本文根据瓦片数据存储组织模型，以瓦片对象为缓存系统的基本存储单元，构造了瓦片数据缓存索引，缓存索引采用 R 树来构建，被缓存的瓦片数据单独存放在缓存文件区域内，其缓存索引在系统启动时预先载入内存中。索引项采用 Key-Value 的结构，Key 为缓存瓦片对象的主键，通过瓦片的空间属性（行号、列号和层级）来确定，可以唯一确定一张瓦片，Value 值为索引对象实体。该缓存索引项结构如图 4.6 所示。每个缓存瓦片数据对应一个索引项，为了保证高并发下的数据安全，使用 CurrentHashMap<Key,Value>的形式存储。

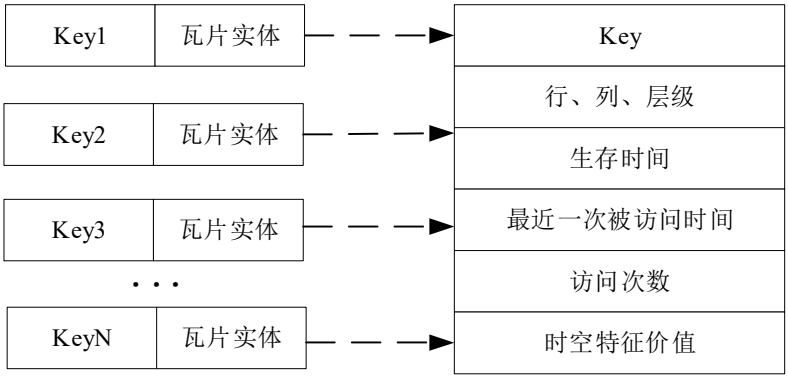

图 4.6 缓存瓦片数据及索引示意图

4.3.3 基于瓦片时空特征价值的缓存置换流程

基于瓦片时空特征价值的缓存置换策略过程如图 4.7 所示,具体实现过程为:

（1）当有新的瓦片数据访问请求且未命中缓存或者有新的瓦片数据加入缓存时，首先需要创建并且初始化缓存对象，$count_{tol}$初始化为 1，t_{last}为当前系统时间，生存期限为 30 天（2592000000L）;

（2）计算新的缓存瓦片对象的时空特征价值，更新缓存空间中与新加入的瓦片缓存对象相关的其它瓦片数据的时空特征价值，即缓存索引信息;

（3）遍历缓存空间中的所有缓存瓦片对象的生存时间是否已过生命周期，将超过生命周期的缓存瓦片对象移除;

（4）判断是否有剩余缓存空间存储新加入的缓存瓦片对象，如果有则直接缓存，否则执行下一步;

（5）将最小时空特征价值的缓存瓦片对象移除掉，判断此时剩余缓存空间是否能够缓存新加入的缓存对象，如果可以则直接缓存，否则继续执行该步骤，直至新的瓦片对象加入缓存。

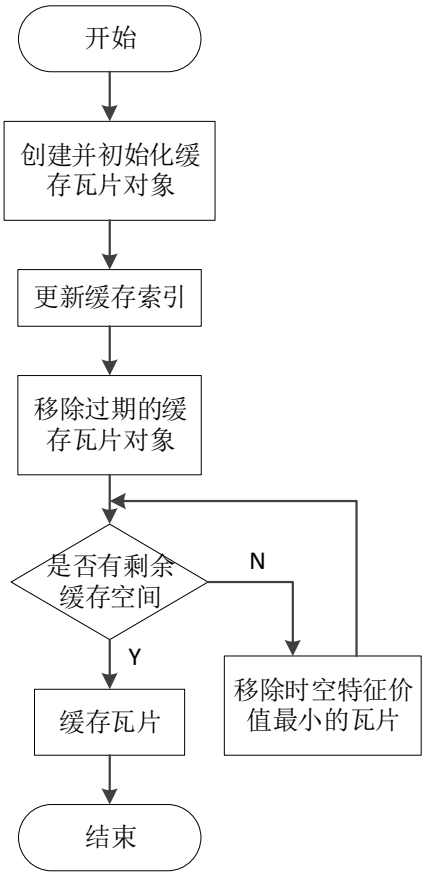

图 4.7 基于时空特征价值的瓦片缓存置换流程

4.4 本章小结

本章主要从遥感影像瓦片的访问特性以及时空特征价值出发，制定了基于深度优先策略和广度优先策略的预取策略，提出了基于时空特征价值的瓦片缓存置换策略，通过减少数据交互次数，有效地缓解了 Ceph 存储系统的网络带宽和服务器压力，提高了瓦片数据的读取性能，降低了用户的访问时间。本章充分利用了遥感影像瓦片独特的时空特征价值，为基于 Ceph 的遥感影像瓦片存储系统提供了瓦片的预取策略和缓存置换策略。

（1）有效的预取策略通过预取部分瓦片数据，减少了后端存储系统的数据交换次数 ，提高了瓦片数据的读取性能和访问效率，同时也可以减少高并发场景下 Ceph 集群的压力，提高用户在地图操作时的体验。

（2）基于遥感影像瓦片时空特征价值的缓存置换策略能够充分发挥缓存服

务器的存储性能，同时该缓存置换策略能够更充分地利用瓦片的时空特性，提高瓦片数据的缓存命中率，从而有效提高存储系统的读性能。

5 原型系统实现与性能测试

前面几章我们分别介绍了 Ceph 分布式存储的关键技术、基于扩展 Z 空间填充曲线和一致性哈希算法设计的遥感影像瓦片合并存储策略以及瓦片数据预取和缓存策略的研究。本章将基于这些关键技术构建遥感影像瓦片存储系统，并使用原型系统对这些关键技术的性能和有效性进行相关测试。

5.1 原型系统实现及部署

本文实验环境由八台服务器构成，其中三台服务器用来搭建 Ceph 分布式存储集群，三台服务器用来搭建瓦片数据合并服务集群，一台服务器作为 Ceph 分布式集群的访问代理，主要提供存储集群的瓦片合并文件读写服务，最后一台服务器作为遥感影像瓦片存储系统的后端服务器，主要提供遥感影像瓦片的上传和下载服务。整个遥感影像瓦片存储系统的实验环境拓扑图如图 5.1 所示，主要由 Ceph 分布式存储集群、遥感影像瓦片合并服务集群、Ceph 集群代理服务器以及遥感影像瓦片存储系统后端服务器组成。下面将对该实验环境拓扑图中的重要组成部分进行介绍。

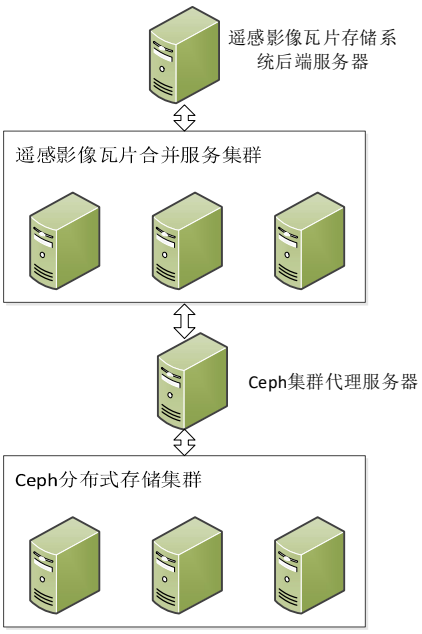

图 5.1 遥感影像瓦片存储系统实验环境拓扑图

（1）Ceph 分布式存储集群

Ceph 分布式存储集群主要负责瓦片合并文件的存储，并提供基于对象存储的数据读写服务。本实验环境中的 Ceph 分布式存储集群共包含 3 个节点，每个节点分别使用 3 块 30GB 的 SATA 硬盘作为 OSD，OSD 底层文件系统采用 Ext4。该存储集群总共包含 9 个 OSD 节点。此外，每个节点服务器启动一个 MON 进程，共使用三台服务器构建 Ceph 存储集群的监视器集群，用于对整个存储系统的健康状况进行监控，其中一台服务器上部署对象网关服务，主要提供数据的读写服务。Ceph 分布式集群组成如图 5.2 所示。

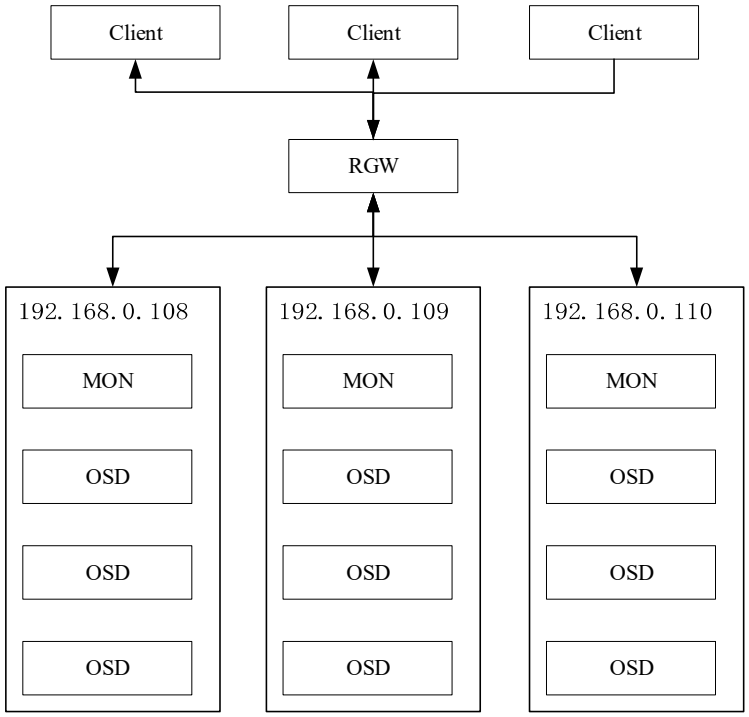

图 5.2 Ceph 集群构建策略

Ceph 分布式存储集群中每个节点服务器的硬件环境通常包括内存、CPU、主板、硬盘等，具体的硬件信息如表 5.1 所示，集群服务器部署软件版本信息如表 5.2 所示：

表 5.1 Ceph 集群服务器硬件环境

硬件	型号
CPU	Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz
内存	2GB
磁盘	3 块 30G SATA 硬盘
主板	440BX Desktop Reference Platform

表 5.2 操作系统及 Ceph 版本

操作系统	CentOS Linux release 7.7.1908 (Core)
Ceph 存储系统版本	ceph version 12.2.13
单节点 OSD 数量	3

（2）遥感影像瓦片合并服务集群

遥感影像瓦片合并服务集群主要提供影像瓦片的合并服务，将空间上彼此相邻的瓦片数据进行合并，并更新瓦片数据到合并文件的全局映射索引和本地索引，最后调用 Ceph 存储集群访问代理提供的合并文件读写接口来完成合并文件的读写。本实验环境中的合并服务集群共包括 3 个合并服务节点，每个合并服务节点部署一个瓦片数据合并服务进程，合并服务节点部署的操作系统以及 JDK 版本如表 5.3 所示：

表 5.3 合并服务节点部署软件版本

操作系统	Windows 10 64 位
JDK	java version 1.8.0_144

（3）Ceph 存储集群访问代理

Ceph 存储集群访问代理主要负责向 Ceph 存储集群提供瓦片合并文件的读写、删除、查询等服务。此外，本实验还使用该代理服务器的 SSD 盘作为瓦片数据文件缓存区，用来提高整个遥感影像瓦片存储系统的读写性能。该访问代理提供的主要服务见表 5.4：

表 5.4 Ceph 存储集群访问代理提供的服务

服务名称	功能描述
writeToCeph	瓦片合并文件写入 Ceph 集群
readMergeTileToCache	从 Ceph 集群读取瓦片合并文件
listObjects	查询 Ceph 集群中的瓦片合并文件
createContainer	在 Ceph 集群中创建瓦片合并文件存储目录
deleteContainer	删除 Ceph 集群中的瓦片合并文件存储目录
deleteAllObjects	删除 Ceph 集群中所有的瓦片合并文件

（4）遥感影像瓦片存储系统后端服务器

遥感影像瓦片存储系统后端服务器主要负责接收用户的瓦片上传和下载请求，并对请求进行相应的预处理（比如瓦片参数解析、根据瓦片名称判断该瓦片是否符合瓦片命名规范以及对瓦片进行 Z 曲线编码等），最后根据一致性哈希算法调用相应合并服务节点提供的瓦片数据合并服务。遥感影像瓦片存储系统后端

服务器提供的主要服务如表 5.5 所示：

表 5.5 遥感影像瓦片存储管理系统后端服务器提供的服务

服务名称	功能描述
upload	上传用户指定目录下的瓦片数据
download	下载瓦片数据到指定目录

5.2 Ceph 存储集群网络配置

Ceph 存储集群中每台服务器分别配置两块网卡，一个是公共网络，使用 192.168.0.X 网段，主要用于对外提供数据读写服务；另一个是 Ceph 存储集群私有网络，使用 192.168.142.X 连接，主要用于存储集群内部的数据传输。服务器具体网络配置信息见表 5.6：

表 5.6 Ceph 存储集群网络部署情况

IP	主机名	组件
192.168.0.108/192.168.142.111	ceph-node1	OSD/MON/RGW
192.168.0.109/192.168.142.112	ceph-node2	OSD/MON
192.168.0.110/192.168.142.113	ceph-node3	OSD/MON
192.168.0.106/192.168.142.115	cluster-proxy	cluster-service

5.3 系统实验验证

本文主要从遥感影像瓦片合并存储策略、缓存置换策略和预取策略三方面对海量遥感影像瓦片多级优化存储方法 TMOSM 的读写性能进行验证。其具体的实验流程如图 5.3 所示。

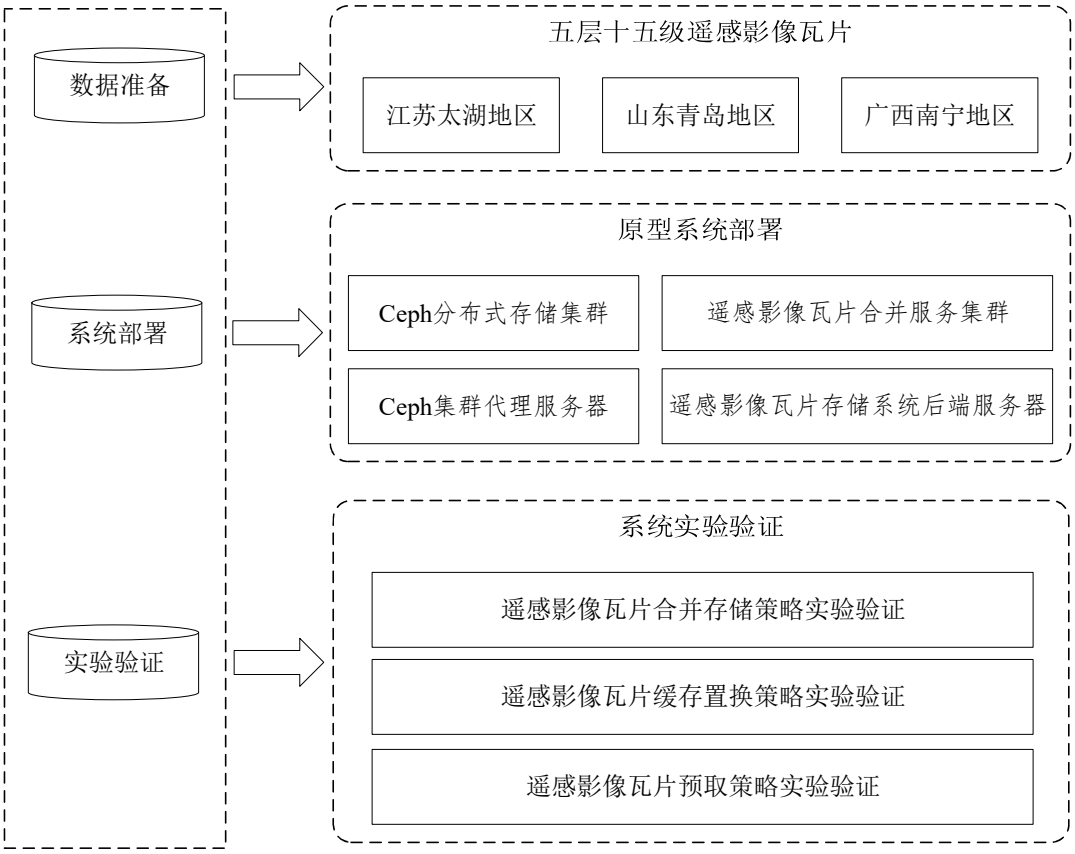

图 5.3 系统实验验证流程图

5.3.1 实验数据与内容

本次系统实验采用的瓦片数据为江苏太湖地区、山东青岛地区和广西南宁地区的五层十五级遥感影像瓦片（中国科学院数字地球与遥感研究所，2019），写入数据总大小为 17.5G，瓦片总数目为 9882 条；读取的瓦片数据总大小为 1.89G，瓦片总数目为 1009 条，其中读取的瓦片数据为写入瓦片数据的一部分。

本次系统实验主要包括三部分，第一部分测试遥感影像瓦片合并存储策略的性能，主要比较采用 TMOSM 前后瓦片数据读写性能的变化；第二部分测试基于瓦片时空特征价值的缓存置换策略的有效性；第三部分测试不同操作类型下瓦片预取策略的性能。

5.3.2 遥感影像瓦片合并存储策略实验验证

遥感影像瓦片合并存储策略实验的目的主要在于比较 Ceph 遥感影像瓦片存储系统采用海量遥感影像瓦片多级优化存储方法 TMOSM 前后瓦片数据读写性

能差别，其实验过程主要是采用原型系统来测试瓦片数据的读写性能，然后通过实验数据比较存储系统采用 TMOSM 前后瓦片数据读写性能的变化。

5.3.2.1 实验内容

本实验分别采用五个线程对 Ceph 遥感影像瓦片存储系统进行并发读写性能测试。Ceph 存储集群中的 OSD 节点底层采用 Ext4 文件系统，为了确保每次实验产生的碎片数据不会对当前实验结果产生影响，每次实验之后都会对 OSD 文件系统进行格式化。此外，为了保证实验结果的准确性，本次实验通过多次测试求平均值来减少由于网络等偶然因素对实验结果产生的影响。

本实验主要分为两个部分，第一部分测试 Ceph 遥感影像瓦片存储系统的瓦片数据写入性能，第二部分测试 Ceph 遥感影像瓦片存储系统的瓦片数据读取性能。遥感影像瓦片存储系统的读写性能测试主要分为六组测试，分别测试 Ceph 系统优化前（合并数据集大小为 1）以及 Ceph 系统优化后（合并数据集大小分别为 40、80、120、500、1000）的瓦片数据读写性能变化，最后比较遥感影像瓦片存储系统采用 TMOSM 前后读写性能差异。本实验每组测试重复进行 3 次，然后求平均值作为最终的实验结果。

5.3.2.2 实验结果与分析

（1）Ceph 遥感影像瓦片存储系统写入性能测试

为了更好地分析 Ceph 遥感影像瓦片存储系统瓦片数据写入性能，本次测试的评价指标包括瓦片数据上传的总时间、瓦片数据合并的总时间、驻留在文件缓存区中的瓦片数目以及瓦片合并文件写入集群的总时间，最终的测试结果分别如图 5.4-5.7 所示。

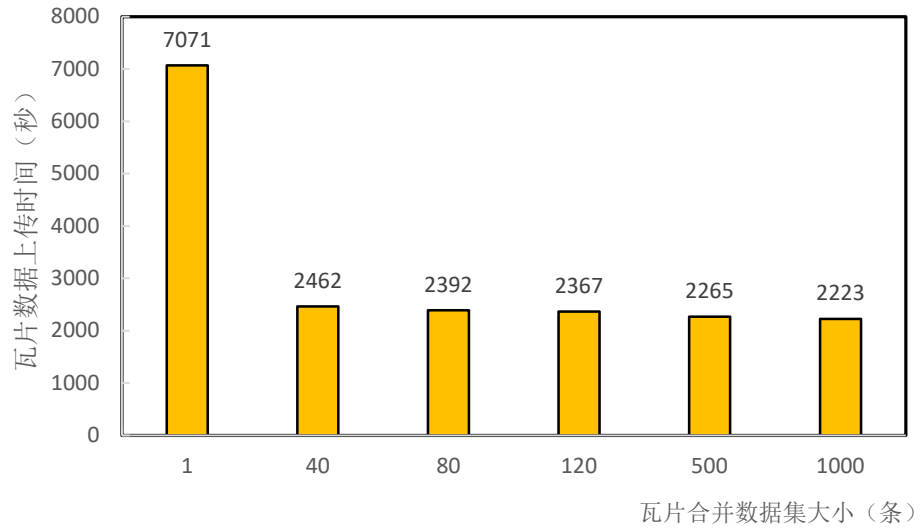

图 5.4 不同数据集大小下瓦片数据上传时间比较

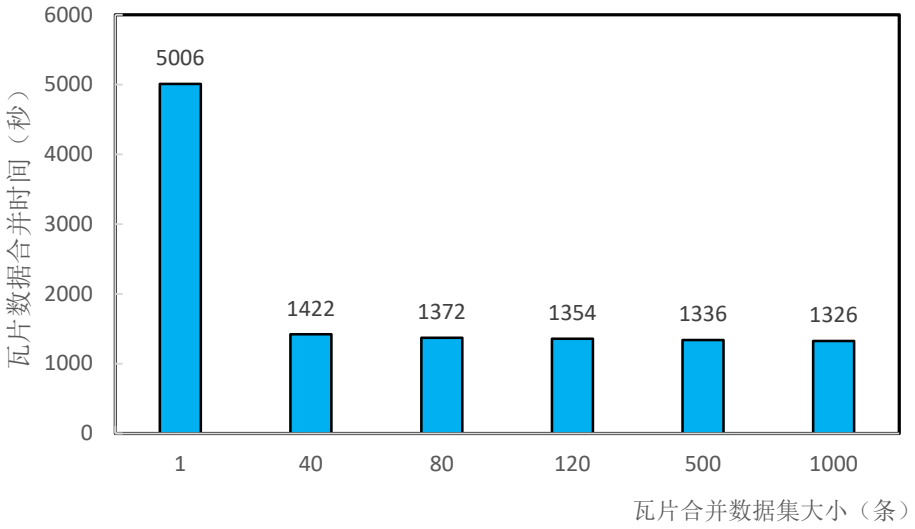

图 5.5 不同数据集大小下瓦片数据合并时间比较

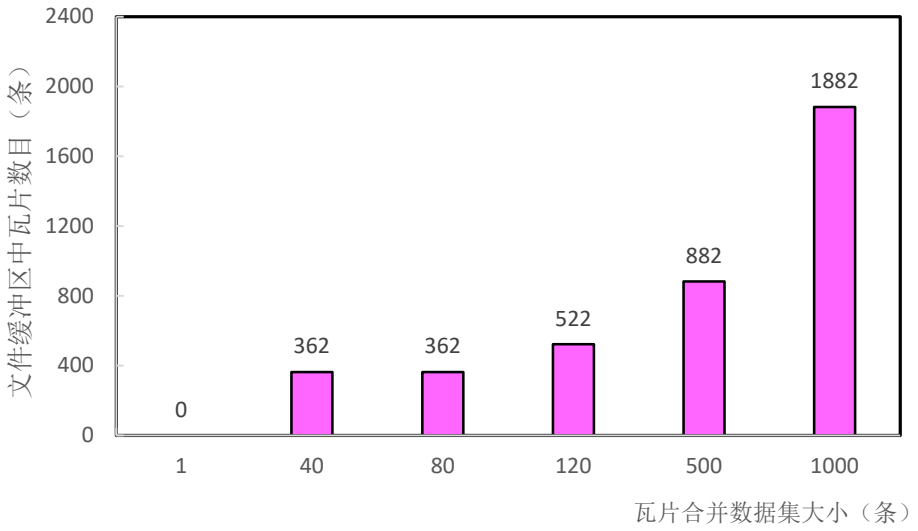

图 5.6 不同数据集大小下文件缓存区中瓦片数目比较

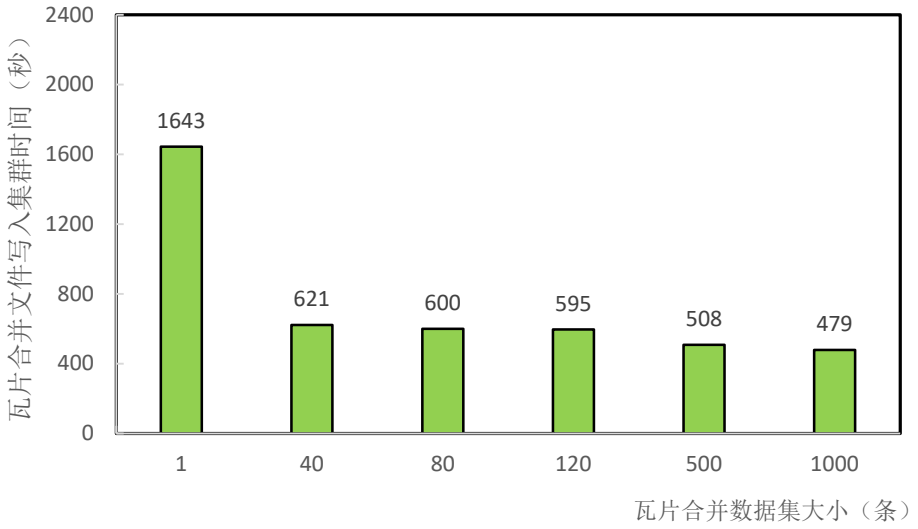

图 5.7 不同数据集大小下瓦片合并文件写入集群时间比较

从图 5.4 中的实验结果可以发现，采用瓦片数据合并存储策略后的存储系统写入性能可以得到明显改善，瓦片数据上传总时间大幅减少，并且随着合并瓦片数目的增多，Ceph 存储系统的写入性能也会有比较明显的提升，具体原因分析如下：

① Ceph 存储系统优化前写入瓦片数据时，会频繁地与 Ceph 存储系统对象网关进行通信，获取瓦片数据的存储地址，然后将瓦片数据写入 Ceph 集群，往往需要更多的网络通信开销以及 CPU 计算。此外，Ceph 集群本身在存储大量小文件时并没有明显的性能优势，因此 Ceph 存储系统优化前直接存储海量瓦片数

据时性能往往表现较差。Ceph 存储系统采用瓦片数据合并存储策略后，将空间上彼此相邻的瓦片数据合并成一个瓦片数据集存入 Ceph 集群，将多个瓦片数据访问合并成一次数据访问，大幅度减少了与 Ceph 集群对象网关的网络通信开销，同时也可以发挥 Ceph 集群在存储大文件时性能表现良好的优势，由图 5.7 的实验结果可以发现，在写入集群瓦片数量基本相同的情况下，Ceph 系统优化后，瓦片数据写入集群的时间大幅度减少，并且随着瓦片数据集的增大，瓦片数据写入集群的时间也会相应减少。

② Ceph 存储系统直接存储海量遥感影像瓦片时需要频繁地创建合并文件，不断地打开和关闭文件输入输出流，从而增加海量瓦片数据的合并时间，Ceph 存储系统优化后中，将多个遥感影像瓦片合并成一个瓦片数据集，减少了瓦片合并文件的创建时间，无需频繁地打开和关闭文件输入输出流。由图 5.5 的实验结果可知，优化后的 Ceph 系统大大减少了瓦片数据的合并时间，并且随着合并瓦片数目的增多，瓦片数据合并时间也会明显减少。

③ 由图 5.6 的实验结果可以发现，Ceph 系统直接存储海量遥感影像瓦片时并没有瓦片驻留在文件缓存区，无法充分发挥文件缓存区的优势，Ceph 系统的写入性能往往表现较差。Ceph 存储系统采用瓦片合并存储策略后，部分瓦片数据会驻留在文件缓存区，同时随着瓦片数据集的增大，驻留在文件缓存中的瓦片数目也会相应增多，文件缓存区的利用率也会相应提高，因此，Ceph 系统的写入性能也会相应改善。但是，如果合并瓦片数据集太大，Ceph 系统的文件缓存区大小将会成为 Ceph 系统的瓶颈，所以往往需要根据文件缓存区的大小合理地设置合并数据集的大小。

（2）Ceph 遥感影像瓦片存储系统读取性能测试

为了更好地分析 Ceph 遥感影像瓦片存储系统瓦片数据读取性能，本次测试的评价指标包括瓦片数据读取的总时间以及瓦片数据缓存命中率，最终的测试结果分别如图 5.8-5.9 所示。

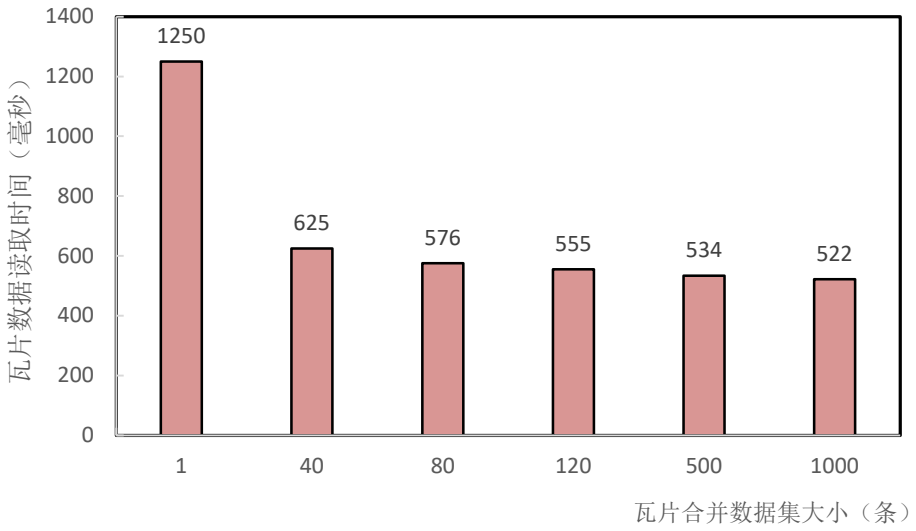

图 5.8 不同数据集大小下瓦片数据读取时间比较

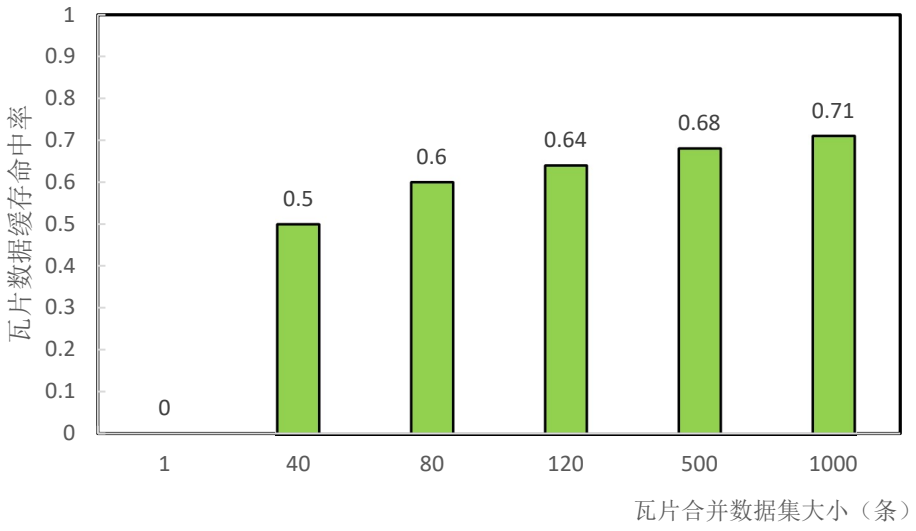

图 5.9 不同数据集大小下瓦片数据缓存命中率比较

从图 5.8 中的实验结果可以发现，采用瓦片数据合并存储策略后的存储系统瓦片数据读取性能可以得到明显改善，瓦片数据读取时间大幅减少，并且随着合并瓦片数目的增多，Ceph 存储系统的读性能也会有比较明显的提升，具体原因分析如下：

① Ceph 系统优化前读取瓦片数据时，会多次与 Ceph 存储系统对象网关进行通信，获取瓦片数据存储位置，然后将瓦片数据读取到指定目录，这个过程往往需要更多的网络通信开销以及 CPU 计算。此外，Ceph 集群本身在读取小文件时无法突出其性能优势。因此，从 Ceph 存储系统读取瓦片数据时往往性能较差。

Ceph 存储系统采用瓦片数据合并存储策略后，读取瓦片数据时会首先在文件缓存区中查找，如果缓存区中存在查找的瓦片，则直接将缓存区中的瓦片读取到指定目录，如果缓存区中不存在，才会向集群中请求待读取的瓦片合并文件，并且会把合并文件中的部分瓦片预取到文件缓存区。因此采用瓦片数据预取和缓存技术后，大量瓦片数据可以直接从文件缓存区中读取，从而提高 Ceph 存储系统的瓦片数据读取性能。

② 由图 5.9 的实验结果可以发现，Ceph 存储系统优化前瓦片数据的缓存命中率为 0，说明用户读取的瓦片直接从 Ceph 集群中读取，无法充分地利用文件缓存区，Ceph 存储系统读性能表现较差。采用遥感影像瓦片合并存储策略后，文件缓存区中的瓦片读取命中率大大提高，并且随着合并瓦片数目的增多，瓦片数据读取缓存命中率明显提高，Ceph 系统的整体读性能也会有所改善。

综上所述，通过遥感影像瓦片合并存储策略实验可以发现，本论文设计的遥感影像瓦片合并存储策略能够有效地提高 Ceph 存储系统的读写性能。

5.3.3 基于时空特征价值的瓦片缓存置换策略的实验验证

5.3.3.1 实验内容

本实验设置的缓存区大小分别为 100MB、200MB、300MB、400MB 以及 500MB。实验采用客户端模拟仿真程序来模拟用户对瓦片地图的操作，主要包括平移、放大和缩小，各操作次数为 1000 次，本实验使用了 LRU、LFU 以及本文提出的基于时空特征价值的缓存置换策略算法三种算法，每种缓存置换策略算法测试 10 次，最后计算出瓦片数据的平均缓存命中率。

5.3.3.2 实验结果与分析

三种缓存置换策略算法对缓存瓦片命中率的影响差异如图 5.10 所示：

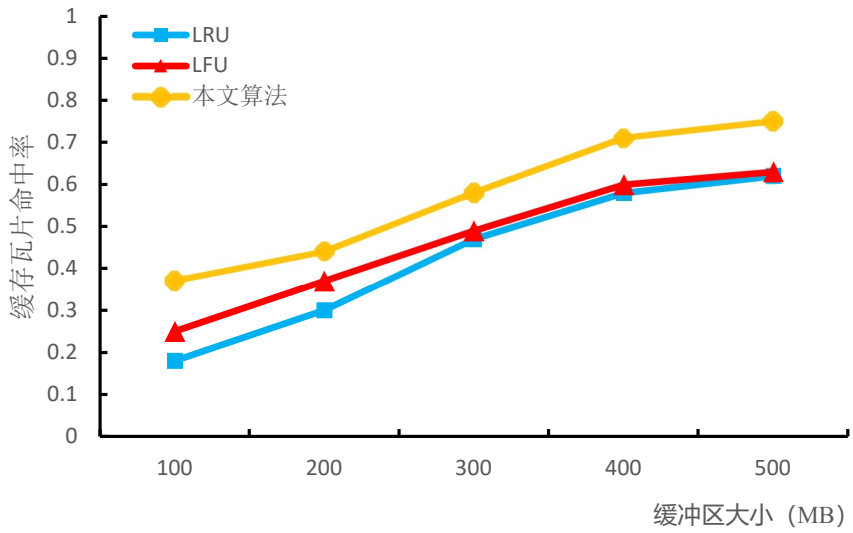

图 5.10 不同缓存置换策略算法下瓦片命中率比较

图 5.10 分别为 LRU、LFU 以及本文提出的基于时空特征价值的缓存置换策略算法的缓存瓦片命中率的实验结果，从实验结果可知，三种瓦片缓存置换策略的缓存瓦片命中率都会随着缓存区大小的增大而提高，当缓存区大小比较小时，三种缓存置换策略算法的缓存瓦片命中率增加较为明显；当缓存区大小比较大时，三种缓存置换策略算法的缓存瓦片命中率增加相对平缓。在缓存瓦片命中率方面，LFU 算法略优于 LRU 算法，本文设计的瓦片缓存置换策略算法略优于 LFU 和 LRU 算法，该算法充分考虑了瓦片数据的时空特征，在缓存瓦片命中率方面可以获得不错的效果。

5.3.4 基于瓦片预取技术的瓦片数据读取性能测试实验

5.3.4.1 实验内容

本实验主要对 Ceph 遥感影像瓦片存储系统的响应时间进行测试，从而说明预取策略的有效性，实验设置的缓存区大小为 500MB。在网络正常的情况下，使用客户端模拟仿真程序，模拟用户对瓦片地图的操作，主要包括平移、放大和缩小，各操作次数为 100 次。实验记录操作的开始时间和完成时间，从而得到操作的响应时间，最后取平均响应时间作为最终的实验结果。

5.3.4.2 实验结果与分析

采用瓦片预取技术后用户执行地图操作的响应时间与未采用预取技术执行地图操作的响应时间比较如图 5.11 所示：

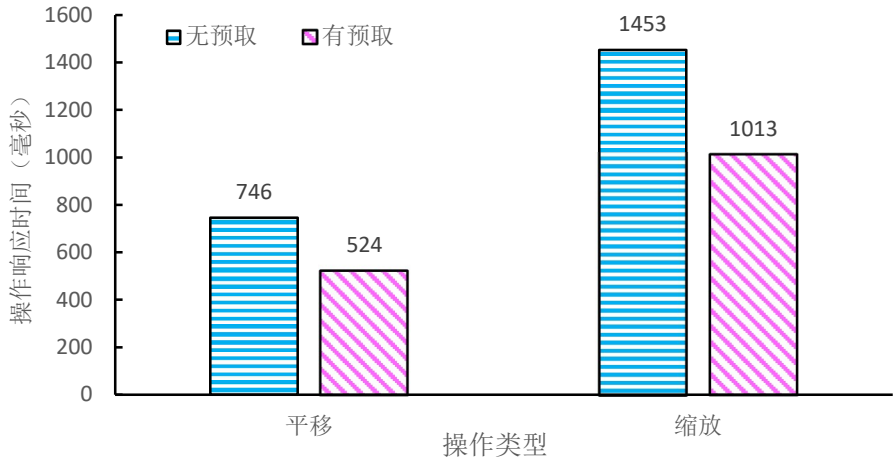

图 5.11 预取策略影响下不同操作类型的响应时间

图 5.11 分别为采用预取策略和未采用预取策略影响下不同操作类型所对应的响应时间。由实验结果可知，采用预取策略后用户在执行平移和缩放操作时的系统平均响应时间有所减少，执行平移操作的平均响应时间减少了 200 多毫秒，执行缩放操作的平均响应时间减少了 400 多毫秒，因此采用预取策略能够在一定程度上提高遥感影像瓦片存储系统的读取性能。

5.4 本章小结

本章主要讨论了遥感影像瓦片存储系统的设计，并详细介绍了组成原型系统的各组件的具体功能及其部署环境。然后基于原型系统，对遥感影像瓦片合并存储策略、基于瓦片数据时空特征价值的缓存置换策略以及瓦片预取策略进行了实验验证，验证了关键技术的有效性。

6 总结与展望

6.1 论文工作总结

随着传感器技术以及通信技术的快速发展，遥感影像的获取方式日益增多，在国土、海洋、农业、林业等多个领域获得了广泛的应用，此外，由于遥感影像切片技术越来越多地应用于遥感影像对外提供的服务中，遥感影像瓦片的规模呈爆炸式增长。传统的基于数据库和文件系统的遥感影像瓦片存储系统，不仅在存储性能方面表现较差，同时也难以扩展和维护，已经无法满足海量遥感影像瓦片的存储，而新一代的基于主从架构的的 NoSQL 数据库以及 HDFS 分布式文件系统，由于存在主节点单点故障问题，其主节点往往容易成为整个存储系统的性能瓶颈。此时，基于无中心化架构的分布式存储系统 Ceph 由于没有单点故障，具有高性能、高可靠性以及高扩展性，已经成为海量遥感影像瓦片存储的最佳选择。此外，Ceph 支持对象存储、文件存储、块存储等多种存储方式，能够满足不同应用场景下遥感影像瓦片的存储需求，但是其在存储遥感影像瓦片等小文件时读写性能往往不能令人满意。因此，本文首先深入研究了开源的分布式文件存储系统 Ceph，然后针对 Ceph 在存储海量遥感影像瓦片等小文件时存在的性能问题，设计并实现了海量遥感影像瓦片多级优化存储方法 TMOSM，该方法包括基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略、基于布隆过滤器和 FNI-Tree 的全局映射索引策略和顾及瓦片时空特性的预取技术和缓存置换策略。最后基于原型系统对这些策略的可行性和有效性进行了验证。本文主要的研究内容如下：

（1）海量遥感影像瓦片多级优化存储方法 TMOSM

针对 Ceph 分布式存储系统在存储海量遥感影像瓦片时读写性能不佳的问题，本文提出一种基于瓦片合并存储与预取缓存的海量瓦片多级优化存储方法（Tile Multilevel Optimal Storage Method，TMOSM）。该方法首先使用扩展 Z 曲线对瓦片数据编码，采用空间降维的方式将空间上彼此邻近的瓦片数据合并成一个大文件存入 Ceph 集群，然后通过构建基于布隆过滤器与 FNI-Tree 的全局映射索引，实现瓦片数据的快速读取，最后根据瓦片数据的访问特性和时空特征价值，预取

并缓存热点瓦片数据，并使用基于瓦片时空特征价值的缓存置换策略，减少节点间的网络通信开销，提高瓦片数据的读取性能。该方法充分的考虑了遥感影像瓦片的时空特性，为海量遥感影像瓦片的分布式存储提供了一种有效方法。

（2）基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略

遥感影像瓦片合并存储策略是提高基于 Ceph 遥感影像瓦片存储系统写入性能的关键技术。为了将具有高度空间相关性的遥感影像瓦片合并成一个大文件，本文在现有数据划分的基础上，充分结合遥感影像瓦片的时空特性以及分布式环境的特征，提出了基于扩展 Z 曲线和一致性哈希的遥感影像瓦片划分策略。该策略包括遥感影像瓦片数据集的构建和遥感影像瓦片数据集的分发。基于扩展 Z 曲线的瓦片编码能够将空间上邻近的瓦片划分到一起，而一致性哈希算法能够根据合并服务节点的性能进行遥感影像瓦片数据集的合理分发。此外，遥感影像瓦片数据集的大小是影响整个存储系统性能的关键因素。实验结果表明，本文提出的遥感影像瓦片合并存储策略能够有效提高瓦片数据的写入性能，并且缓存空间足够的情况下，随着遥感影像瓦片数据集的增大，其瓦片数据写入性能也在不断提高。

（3）顾及瓦片时空特性的预取技术和缓存置换策略的实现

本文从遥感影像瓦片的访问特性以及时空特征价值出发，将预取和缓存技术应用于遥感影像瓦片存储系统服务中。为了充分利用用户在地图浏览时的停顿时间，制定了基于广度优先策略和深度优先策略的预取策略。此外，充分结合遥感影像瓦片的访问特性和时空特征价值，提出了基于时空特征价值的瓦片缓存置换策略。实验结果表明，顾及瓦片时空特性的预取技术和缓存置换策略能够有效提高遥感影像存储系统响应速度以及遥感影像瓦片的读取性能。

（4）遥感影像瓦片存储系统原型系统设计与实现

基于上述关键技术，本文设计并构建了遥感影像瓦片存储系统原型系统，并采用 dubbo 框架、Spring、Ceph Java API 等相关技术对原型系统进行了具体实现，最后对上述关键技术的可行性和有效性进行了实验验证。实验结果表明，基于上述关键技术构建的遥感影像瓦片存储系统不仅具有高可用性、高可靠性以及高扩展性，而且有效解决了 Ceph 存储系统直接存储瓦片数据时读写性能不佳的问题。

6.2 研究特色

本文的研究特色主要体现在以下三个个方面：

（1）设计了海量遥感影像瓦片多级优化存储方法 TMOSM

为了解决 Cpeh 存储系统在存储海量遥感影像瓦片等小文件时读写性能不佳的问题，本文设计了基于遥感影像瓦片合并存储和预取缓存的海量遥感影像瓦片多级优化存储方法，该方法结合基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略、基于布隆过滤器和 FNI-Tree 的全局映射索引策略和顾及瓦片时空特性的预取技术和缓存置换策略等多种策略从多个方面提高基于 Ceph 的遥感影像瓦片存储系统读写性能。

（2）提出了基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略

为了充分利用遥感影像瓦片的空间特性，将空间上彼此邻近的瓦片合并在一起存入 Ceph 系统，同时考虑到瓦片合并服务集群的可扩展性，本文提出了基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略，有效地改善了遥感影像瓦片的写入性能，同时也使得遥感影像存储系统具有更好的伸缩性。

（3）提出了顾及时空特性的瓦片预取技术和缓存置换策略

为了充分利用用户在进行地图操作时的停顿时间，本文采用预取技术根据用户的访问特性将用户执行下一步操作时可能访问的瓦片数据提前预取到服务器缓存空间，通过减少数据交换次数，降低网络通信开销，提高遥感影像存储系统的响应时间。此外，设计了基于瓦片时空特征价值的缓存置换策略，充分利用了遥感影像瓦片的时空特性和访问特性，缓解了 Ceph 存储系统服务器压力，有效地改善了 Ceph 遥感影像瓦片的读取性能。

6.3 不足与展望

本文对 Ceph 分布式存储系统进行了深入研究和分析，并结合遥感影像瓦片的时空特性，设计了包含基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略，基于布隆过滤器和 FNI-Tree 的全局映射索引策略和顾及瓦片时空特性的预取技术和缓存置换策略的海量遥感影像瓦片多级优化存储方法 TMOSM，有效改善了 Ceph 遥感影像瓦片存储系统的读写性能。但是，针对 Ceph 存储海量

遥感影像瓦片性能不佳的问题还可以进一步的研究，其深入研究的方向包括如下几个方面：

（1）本文设计的基于扩展 Z 曲线和一致性哈希的遥感影像瓦片合并存储策略能够将空间上相邻的瓦片数据合并在同一文件中，但是并未考虑瓦片数据的其它属性，如卫星、传感器、生产时间等非空间属性，因此结合具体的应用场景设计更好的相关性分类算法，能够将相关性更强的瓦片数据合并在一起，从而提高缓存区中瓦片数据的命中率。

（2）本文设计的海量遥感影像瓦片多级优化存储方法 TMOSM 与瓦片合并数目存在着密切的关系，但是未对瓦片合并数目的阈值进行探究，后续的工作可以针对瓦片合并数目阈值进行探索，使得遥感影像瓦片存储系统在读写性能、缓存命中率、系统响应时间等方面达到最优。

（3）本文设计的海量遥感影像瓦片多级优化存储方法 TMOSM 虽然能够提高海量瓦片数据的读写性能，但是也会造成一定的网络通信开销。后续的工作可以基于 Ceph 分布式存储系统源代码，深入分析 Ceph 存储系统机制，从源码层面设计更好的瓦片数据合并方案，这样可以更加有效地提升 Ceph 存储系统的读写性能。

参考文献

曹梦鸽，高心丹，程逸群. 基于 HBase 的森林防火遥感瓦片大数据存储[J]. 东北林业大学学报. 2018,46(2): 35-39.

程昌秀. 空间数据库管理系统概论[M]. 2012.

陈时远. 基于 HDFS 的分布式海量遥感影像数据存储技术研究[D]. 中国科学院大学(工程管理与信息技术学院), 2013.

褚信，蔡阳军，杜震洪，等. 用户行为选择参与的五层十五级瓦片缓存置换策略研究[J]. 浙江大学学报(理学版). 2016, 43(04): 452-457.

董献伦. 基于关系型数据库的数据切分问题研究[D]. 山东大学, 2016.

何中林. 集群的可扩展性及其分布式体系结构分析与研究[J]. 乐山师范学院学报(5):71-74.

胡庆武, 陈亚男, 周洋, 等. 开源 GIS 进展及其典型应用研究[J]. 地理信息世界, 2009, 7(1): 46-55.

姜斌. 高分遥感数据的多领域专题应用定制化开发模型研究[D]. 浙江大学, 2018.

李振举，李学军，谢剑薇，等. 基于 HBase 的海量地形数据存储[J]. 计算机应用. 2015(07):1849-1853.

刘爱贵. 海量小文件问题综述 [EB/OL]. http://blog.csdn.net/liuaigui/article/details/9981135?utm_source=tuicool&utm_medium=referral，2013-08-15.

刘高军, 王帝澳. 基于 Redis 的海量小文件分布式存储方法研究[J]. 计算机工程与科学, 2013, 35(10).

刘佳星，陈飞翔，陈星涵. 一种基于地理单元热度的瓦片缓存策略[J]. 计算机工程与应用. 2017,53(05): 90-96.

廖鑫, LIAOXin. 一种基于 LRU 算法改进的缓存方案研究与实现[J]. 信息化研究, 2008(7).

卢秉亮, 梅义博, 刘娜. 位置相关查询中基于最小访问代价的缓存替换方法[J]

吕雪锋, 程承旗, 龚健雅, 等. 海量遥感数据存储管理技术综述[D]. , 2011.

马卫春. MongoDB 存储地图瓦片技术在国情普查建库中的应用[J]. 测绘地理信息. 2018(3).

聂云峰, 周文生, 舒坚, et al. 基于 Z 曲线的瓦片地图服务空间索引[J]. 中国图象图形学报, 2012, 17(2):286-292.

商秀玉. WebGIS 海量瓦片数据管理引擎的设计与实现[D].

宋欣. 基于开源 GIS 软件的 WebGIS 系统构建及应用研究[D]. 兰州: 兰州交通大学, 2012.

孙家柄. 遥感原理与应用[J]. M], 武汉大学出版社, 2003, 21.

孙忠芳，谭智，付海龙. 基于 MongoDB 集群的高性能瓦片服务器[J]. 中国科技信息. 2015(8):107-109.

涂振发，孟令奎，张文，等. 面向网络 GIS 的最小价值空间数据缓存替换算法研究[J]. 华中师范大学学报(自科版). 2012, 46(2): 230-234.

王浩, 潘少明, 彭敏, et al. 数字地球中影像数据的 Zipf-like 访问分布及应用分析[J]. 武汉大学学报(信息科学版)(3):108-111.

王浩, 喻占武, 曾武, et al. 基于瓦片寿命和访问热度的海量空间数据缓存置换策略%Massive Spatial Data Cache Replacement Policy Based on Tile Lifetime and Popularity[J]. 武汉大学学报：信息科学版, 034(006):667-670.

杨彧剑, 林波. 分布式存储系统中一致性哈希算法的研究[J]. 电脑知识与技术(22):29-30.

原发杰. 一种新的海量遥感瓦片影像数据存储检索策略[D]. 成都: 电子科技大学, 2013.

喻凯, 熊祥瑞, 高涛. 基于 Hadoop 的地图瓦片云存储系统的设计与实现[J]. 测绘地理信息(3):78-81.

张毕涛. 分布式存储系统小文件性能优化方案的设计与实现[D]. 北京邮电大学, 2016.

张飞龙. 基于 MongoDB 遥感数据存储管理策略的研究[D]. 河南大学, 2016.

张剑波，夏灯城，赵加奥，等. 分布式计算环境下的栅格数据存储策略[J]. 国防科技大学学报. 2017,39(6): 51-58.

张可力. 面向分布式数据库的瓦片查询热点的缓存系统设计与实现[D]. 华中科技大学, 2014.

张晓兵. 基于 HBase 的弹性可视化遥感影像存储系统[D]. 浙江大学, 2016.

张震波, 杨鹤标, 马振华,等. 基于 LRU 算法的 Web 系统缓存机制[J]. 计算机工程, 2006, 32(19):68-70.

赵晓永, 杨扬, 孙莉莉, et al. 基于 Hadoop 的海量 MP3 文件存储架构[J]. 计算机应用(6):240-242.

赵洋. 淘宝 TFS 深度剖析. 数字化用户, 2013, (3): 58~59.

赵跃龙, 谢晓玲, 蔡咏才, et al. 一种性能优化的小文件存储访问策略的研究[J]. 计算机研究与发展, 2012, 49(7):1579-1586.

郑传建. Ceph 对象文件系统添加任务迁移特性的研究[D]. 武汉理工大学, 2014.

谢毅. 海量遥感影像数据存储组织结构研究[D]. 2011.

周恩强, 董勇, 张伟,等. 对象存储并行文件系统小文件性能优化研究[J]. 计算机工程与科学, 2013, 35(12):8-13.

周林. OpenStack 和 Ceph 结合的云存储设计与实现[D]. 南京邮电大学, 2018.

Barclay T, Chong W, Gray J. “TerraServer Bricks–a high availability cluster alternative.” Microsoft Technical Report MSR-TR-2004-107,2004

Barclay T, Gray J, Slutz D. Microsoft TerraServer: a spatial data warehouse. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. Dallas, Texas: ACM, 2000. 29(2): 307–318.

Beaver D, Kumar S, Li H C, et al. Finding a Needle in Haystack: Facebook's Photo Storage[C]//OSDI. 2010, 10(2010): 1-8.

Bell D G, Kuehnel F, Maxwell C, et al. NASA World Wind: opensource GIS for mission operations. In: Proceedings of IEEE Aerospace Conference. Big Sky, MT: IEEE, 2007. 1–9.

Chang F , Dean J , Ghemawat S , et al. Bigtable: A Distributed Storage System for Structured Data[J]. ACM Transactions on Computer Systems, 2008, 26(2):1-26.

Cloud I B M. Gluster Docs,“[J]. GlusterFS Documentation”, GlusterFS, URL: http://gluster. readthedocs. io/en/latest/Hämtad, 2016: 05-10.

Dean J, Ghemawat S. MapReduce: simplied data processing on large clusters. Commun ACM, 2008, 51(1): 107–113.

Device C B. Ceph Documentation[J]. INKTANK STORAGE, Inc. Ceph [online], 2014.

Fumito Yamaguchi, Hiroaki Nishi. Hardware-based hash functions for network applications[C]// 2013 19th IEEE International Conference on Networks (ICON). IEEE, 2013.

Gang J Z, Zhang G, Tai S Y. Intelligent prefetch and cache techniques based on network preformance [J]. Journal of Computer Research & Development, 2001.

Ghemawat S, Gobioff H, Leung S T. The Google file system[J]. 2003.

Ghemawat S, Gobioff H, Shun-Tak L. The Google file system. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP'03). Bolton Landing, New York: IEEE, 2003. 1–15.

Gorelick N, Hancher M, Dixon M, et al. Google Earth Engine: Planetary-scale geospatial analysis for everyone[J]. Remote Sensing of Environment, 2017, 202: 18-27.

Gremillion, Lee L. Designing a Bloom filter for differential file access[J]. Communications of the Acm, 1982, 25(9):600-604.

Guo W, Gong J Y, Jiang W S, et al. OpenRS-Cloud: a remote sensing image processing platform based on cloud computing environment.Sci China Tech Sci, 2010, 53(suppl. 1): 221–230.

Hadoop Archives Guide [EB/OL]. https://hadoop.apache.org/docs/r1.2.1/hadoop_archives.html，2013-04-08.

Hsiao C C, Chu S L, Dai S S. Efficient rendering and cache replacement mechanisms for hierarchical tiling in mobile GPUs[C]// 2013.

Jayakar K P,Gurav Y B.Managing Small Size Files through Indexing in Extended Hadoop File System [J].International Journal of Advance Research in Computer Science an Management Studies,2014,2(8):161-167.

Kang Y K, Kim K C, Kim Y S. Probability-Based Tile Pre-fetching and Cache Replacement Algorithms for Web Geographical Information Systems[M]// Advances in Databases and Information Systems. 2001.

Korat V G,Pamu K S.Reduction of Data at Namenode in HDFS using harballing Technique [J]. International Journal of Advanced Research in Computer Engineering&Technology,2012,1(4):635-642.

Li R, et al. Replacement Method based on Access Spatiotemporal Locality in a Heterogeneous Distributed Cluster-based Caching System for WebGIS[J]. The Open Cybernetics & Systemics Journal.2015, 9(1): 663-668.

Liu X,Peng C,Yu Z.Research on the Small Files Problem of Hadoop [A]. //International Conference on Education,Management,Commerce and Society [C],Atlantis Press,2015:39-43.

Li X, Dong B, Xiao L, et al. Small Files Problem in Parallel File System[C]// 2011.

Lu X, Wasi-Ur-Rahman M, Islam N S, et al. A Micro-benchmark Suite for Evaluating Hadoop RPC on High-Performance Networks[M]// Big Data Benchmarks, Performance Optimization, and Emerging Hardware. 2013.

Niu N, Li C, Guo L. The application and research of geoprocessing service in WebGIS spatial analysis[C]//2013 21st International Conference on Geoinformatics. IEEE, 2013: 1-4.

Pendleton C. The World cccording to Bing. IEEE Comp Gra Appl, 2010, 30(4): 15–17.

Qun Ren M H D. Using semantic caching to manage location dependent data in mobile computing, in: MobiCom ’00[C]// 2000.

Redkar T. Windows Azure Platform. New York: Apress, 2009. 53–104.

Sage A. Weil, Scott A. Brandt, Ethan L. Miller,等. Ceph: A Scalable, High-Performance Distributed File System[C]// 7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA. USENIX Association, 2006.

Sequence File [EB/OL]. https://hadoop.apache.org/docs/r2.6.2/api/org/apache/hadoop/io/SequenceFile.html.

Shengru Tu, Xiangfeng He, Xuefeng Li,等. A Systematic Approach to Reduction of User-Perceived Response Time for GIS Web Services[C]// ACM-GIS 2001, Proceedings of the Ninth ACM International Symposium on Advances in Geographic

Information Systems, Atlanta, GA, USA, November 9-10, 2001. ACM, 2001.

Shvachko K , Kuang H , Radia S , et al. The Hadoop Distributed File System[C]// 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). IEEE, 2010.

Singh K.Learning Ceph [M]. 1th ed. Birmingham:Packt Publishing Ltd,2015:40-57.

Slutz T B J G. Microsoft TerraServer: A Spatial Data Warehouse[J]. Acm Sigmod Record. 2000,29(2): 307-318.

Wang F, Oral S, Shipman G, et al. Understanding lustre filesystem internals[J]. Oak Ridge National Laboratory, National Center for Computational Sciences, Tech. Rep, 2009.

Xuhui Liu, Jizhong Han, Yunqin Zhong. Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS[C]// Proceedings of the 2009 IEEE International Conference on Cluster Computing, August 31 - September 4, 2009, New Orleans, Louisiana, USA. IEEE, 2009.

Yang C T, Lien W H, Shen Y C, et al. Implementation of a software-defined storage service with heterogeneous storage technologies[C]//2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops. IEEE, 2015: 102-107.

Yingwei L , Xiaolin W , Zhuoqun X . Design of a framework for multi-user/application oriented WebGIS services[C]// Computer Networks and Mobile Computing, 2001. Proceedings. 2001 International Conference on. IEEE, 2001.

.

作者简介

作者简历

曹晓裴，男，1993 年出生，山西太原人。2013 年 9 月至 2017 年 6 月就读于武汉大学地理国情监测专业，获工学学士学位。2017 年 9 月至 2020 年 6 月就读于浙江大学地球科学学院，攻读地质工程专业硕士学位。

参与科研项目

国家民用空间基础设施陆地观测卫星共性应用支撑平台项目（项目编号：US2.474.6632KFS）。