基于软件定义网络和多属性决策的 Ceph 存储系统节点选择方法

王 勇[1),2)] 叶 苗[2),3)] 何 倩[1),2)] 郇宜鸣[2)] 康文杰[1)]

[1)](桂林电子科技大学计算机与信息安全学院 广西 桂林 541004)

[2)](桂林电子科技大学认知无线电与信息处理省部共建教育部重点实验室 广西 桂林 541004)

[3)](桂林理工大学信息科学与工程学院 广西 桂林 541004)

摘 要 云存储中的 Ceph 分布式文件系统以其开原性和提供统一存储能力的特点在企业和科研领域得到广泛关注和应用. CRUSH 算法是 Ceph 分布式文件系统中的伪随机数据分布算法,能实现在异构大规模层级结构化存储集群中高效分布数据对象及其副本. 经典 Ceph 云存储系统中在副本模式下选择存储节点时该系统仅以节点存储容量作为唯一选择条件,并没有考虑到网络和节点的负载状况,这影响了系统在网络性能差和节点高负载的情况下的读写性能. 因此,在 CRUSH 算法中加入网络状态和节点负载的测量对提高负载均衡性具有非常重要的作用. 但在传统的网络构架中要获取网络状态需要繁琐的配置和大量的测量开销. 为解决这些问题,该文设计了基于软件定义网络技术的 Ceph 云存储系统模型和存储节点选择策略,首先利用软件定义网络技术实时获取网络和负载状况,以简化网络配置和减小测量开销,然后通过建立并求解出综合考虑了多种因素的多属性决策数学模型来确定存储节点位置. 通过在实际环境中对设计的存储节点选择方法进行读写操作的测试,测试结果表明,与现有的 CRUSH 算法相比,提出的存储节点选择方法可以在保持与原有 Ceph 系统相同的写操作性能的同时,读小文件操作时的吞吐量和读大文件的响应时间得到明显改善.

关键词 软件定义网络;Ceph 存储系统;多属性决策;副本模式;权重因子

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2019.00323

A New Node Selecting Approach in Ceph Storage System Based on Software Defined Network and Multi-attributes Decision-making Model

WANG Yong[1),2)] YE Miao[2),3)] HE Qian[1),2)] HUAN Yi-Ming[2)] KANG Wen-Jie[1)]

[1)](*School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, Guangxi* 541004)

[2)](*Key Laboratory of Cognitive Radio and Information Processing, Guilin University of Electronic Technology, Guilin University of Electronic Technology, Guilin, Guangxi* 541004)

[3)](*Information Science and Technology, Guilin University of Technology, Guilin, Guangxi* 541004)

Abstract The traditional storage model cannot cope with massive data storage capacity scalability, data reliability and high performance, cloud storage systems came into being under this background. Cloud storage systems use distributed file systems and other technologies to assemble different storage devices into a pool of resources through network connections, unified provision of storage services, with high scalability, high reliability and so on. There are many kinds of distributed file systems for cloud storage, Ceph distributed file system, with its open source nature, and providing

收稿日期:2017-11-06;在线出版日期:2018-09-06. 本课题得到国家自然科学基金项目(61662018,61661015,61831013)、中国博士后科学基金项目(2016M602922XB)、广西创新驱动发展专项(科技重大专项桂科 AA18118031)、桂林理工大学科研启动基金项目(GUTQDJJ20172000019)资助.**王 勇**,男,1964 年生,博士,教授,博士生导师,主要研究领域为云计算、网络流量分类、信息安全等. E-mail: ywang@guet.edu.cn. **叶 苗**(通信作者),男,1977 年生,博士,教授,硕士生导师,主要研究领域为网络计算、无线传感器网络、模式识别与图像处理. E-mail: ym@mail.xidian.edu.cn. **何 倩**,男,1979 年生,男,博士,教授,博士生导师,主要研究领域为分布式计算、软件定义网络. **郇宜鸣**,男,1993 年生,硕士研究生,主要研究方向为云计算、分布式存储. **康文杰**,男,1992 年生,硕士研究生,主要研究方向为云计算、分布式存储.

uniform storage capability, has been widely concerned in enterprises and scientific research fields. With the characters of open source and providing uniform storage capability, the Ceph storage system has been widely concerned in scientific research fields and industry application as one of the most regular cloud storage system. Data distribution strategy is a key technology in distributed file system, which determines location of data storage, load balancing and fault tolerance of the system. CRUSH algorithm is a pseudo-random algorithm for data distribution in Ceph distributed file system which can distribute data objects and their replicas efficiently in large-scale and heterogeneous hierarchical structured storage clusters. However, the open-source Ceph storage system uses storage capacity as the sole consideration for selecting storage nodes in replication scheme in its CRUSH algorithm. It ignores the loads on both the network and individual nodes and negatively affects the system's read and write performance under heavy loads or poor conditions. It is important to utilize the network state information and node load in CRUSH algorithm to improve the load balance. But in the traditional network architecture, it needs cumbersome configuration and much of the measurement overhead. To address these deficiencies, we propose a Ceph enhancement that incorporates software-defined network (SDN) abstraction and an improved strategy for storage node selection. First the nodes' and network's load status are obtained via SDN to simplify the network configuration and alleviate the measurement overhead. Compared with the traditional network architecture, getting network state requires cumbersome configuration and a lot of measurement overhead, Software Defined Network(SDN) separates the control plane and the data plane phase. Through the centralized control plane, it Simplifies network measurement and management and provides a flexible and efficient maintenance strategy, which are adopted the SDN technology to complete the monitoring of network and node load. Second, we establish a multi-attributes decision-making model to select storage nodes optimally. It aims to solve the load unbalanced problem of storage node caused by the storage capacity as the constraint condition in CRUSH algorithm. An improved CRUSH algorithm is proposed to add the factors of network state and load in node weight factor and the determination of weight factor has a finer granularity. We tested the performance of our proposed model in a live environment. The results indicated that the designed model and strategy can significantly improved the throughput for small files and response times for reading large files while offering write performance similar to the unmodified Ceph storage system compared with the original CRUSH algorithm.

Keywords software defined network; Ceph storage; multi-attribute decision-making; replication scheme; weighting factor

1 引 言

随着社交网络、物联网以及云计算等技术的快速发展,网络中的数据量在不间断地、呈指数级地增长,传统的存储模式已经无法应对海量数据的存储,企业和科研人员在不断地提出新的替代技术.分布式存储系统[1]通过大量的普通服务器互联,对外作为一个整体提供存储服务,具有可扩展、高可靠性、高性能以及低成本特性.Weil 等人在攻读博士期间设计了一种可以作为分布式存储的分布式文件系统 Ceph[2],以其开源特性以及提供块存储、对象存储和文件系统的统一存储能力,在越来越多的企业中得到应用.

Ceph 作为大规模分布式存储系统,由成千上万的存储节点构成,节点之间通过网络互连.当前,网络成为制约性能的瓶颈因素之一,然而在 Ceph 的设计中并没有把底层网络状况和节点负载考虑在内,数据选择存储位置仅仅是以节点的存储容量作为权重因子[3],存储节点网络负载加重时(比如集群内部进行数据迁移)会占用大量网络带宽,这会使得集群的性能会明显下降,降低系统的读写性能.如果

能实时获取集群中存储节点的网络状态和负载情况并作为节点选择中的重要因素，在副本模式下客户端读取文件会优先选择性能最佳的副本存储节点，集群的读操作性能会得到改善.

要实现以上提到分布式存储系统中存储节点的网络状态信息，需要高效的完成对网络信息的测量工作.传统的网络状态测量方式需要繁琐的配置和大量的测量开销[4-5].软件定义网络(Software Defined Network，SDN)[6-7]作为一种新的网络模式，主要思想是控制平面和数据平面相分离，打破封闭的垂直体系，并引入对网络编程的能力，通过全局的视角增强对整体系统的管控，显著提升了大规模集群存储中数据的传输、控制和管理效率.如果将基于 SDN 的网络测量技术运用到分布式存储系统的网络状态信息测量中，可以用更小的硬件配置和测量开销，达到优化 Ceph 云存储系统的目的.

因此，本文在基于传统的 Ceph 分布式文件系统的基础上，提出了 Ceph 存储系统与软件定义网络结合的系统架构.Ceph 集群部署在软件定义网络的环境中，Ceph 监控节点调用 SDN 控制器收集网络状态和负载情况并将其作为选择存储节点的权重因子，对于多个权重因子利用多属性决策方案计算出节点的权重.副本模式下，集群根据权重选择数据存放的主次节点.本文的主要贡献点如下：(1)在 Ceph 存储节点的权重因子中添加了通过软件定义网络感知的网络状态和负载的因素，权重因子的确定具有更细的粒度；(2)在保持与原有的 Ceph 集群的写操作性能的同时，提升了读操作的性能.

2　相关工作

云存储技术利用分布式存储技术通过有效的映射机制来建立数据与存储设备之间的映射关系，并实现存储系统的数据管理.根据存储系统中对数据对象以及副本的管理和组织方式可以分为：基于元数据服务器的组织结构和无元数据服务器的组织结构[8].

基于元数据服务器的组织结构[9]通过一个或若干个元数据服务器(Meta-Data Server，MDS)记录数据对象的元数据信息，如数据对象的权限、映射关系等.主流的基于元数据服务器的组织结构的存储系统有 Google 文件系统(Google File System，GFS)[1]和 Hadoop 文件系统(Hadoop Distributed File System，HDFS)[10]等.这种方式实现简单、维护容易但容易存在单点性能瓶颈与故障安全问题.分布式元数据管理解决了集中式管理的问题，但同时带来更多的管理开销.

无元数据服务器的组织结构采用去中心化思想，避免了引入元数据服务器带来的问题，存储位置经过一定的算法计算得到.与前者相比，无元数据的设计实现了系统线性可扩展的能力.主流的无元数据服务器的组织结构的存储系统有分布式文件系统 Ceph[2]、OpenStack Swift 对象存储系统[11，12]、Dynamo 存储系统[12-13]、GlusterFS 分布式文件系统[14]、Ceph 存储系统[2-3].

在无元数据服务器的大规模云存储系统中数据布局算法占有举足轻重的地位.目前广泛使用的算法有一致性 HASH 算法[11]、弹性 HASH 算法和 CRUSH(Controlled Replication Under Scalable Hashing)算法[15].国内也有许多有关云存储系统的数据布局算法方面的研究，国防科技大学的陈涛等人[16]提出了 CCHDP 算法，将聚类算法与一致 HASH 方法相结合，在引入少量的虚拟设备情况下减少了存储空间，同时减少了定位数据消耗的时间.中科院的黄秋兰等人[17]提出一种利用“二分”的映射方式映射物理存储节点，实现节点的快速选择.但这些节点选择算法在考虑数据位置映射关系时没有结合考虑网络负载状况.考虑到 Ceph 云存储系统开源的特性，本文所做工作为对 Ceph 云存储系统中的 CRUSH 算法，结合考虑网络负载信息设计出物理存储节点的数据映射方法，以提高分布式存储系统的性能.

目前在 Ceph 存储数据分布方面，Gudu 等人指出通过将 Ceph 中的 Journal 移至更快的内存中可有效增加吞吐量[18]，同时在 Ceph 存储集群随着客户端增多引起性能下降时，可通过加大 RAM 和数据块大小提升其系统性能.Zhang 等人将 Ceph 与 OpenStack 相结合，并用一些标准测试工具对其进行性能测试，结果显示 Ceph 分布式存储系统部署在 OpenStack[19]云中性能良好并且具有一定的扩展性[20].在处理 Ceph 分布式存储系统的读写负载方面，早在 Ceph 刚问世的 2008 年，加州大学的学者 Esteban 等就对 Ceph 的读写负载处理引入了均衡权重，主次读写方式主动切换等主要方法，对后来的研究有着很强的借鉴意义[21].近几年来，在针对 Ceph 工作负载方面的研究也有很多，Edwin 等人提出了一种基于 MapReduce 的分治策略，保证了 Ceph 存储节点工作量负载的平衡，有效地解决了系统在处理较大数据量时计算应用迁移的问题[22].加州大学圣克鲁兹分校的 Sevilla 等人也提出了针对 Ceph 文件系统负载处理策略[23].这些数据存储位

置选择策略和工作负载方面的讨论没有结合存储网络状态信息方面的考虑.

传统的网络状态信息的测量和获取方法需要繁琐的网络配置,开销大,这就要求必须找到一种灵活方便的网络信息测量方法,才有可能实现在分布式存储系统中结合网络因素对存储节点位置进行选择的方法.软件定义网络(SDN)作为一种新的网络模式,采用控制平面和数据平面相分离,控制层面向集中化和统一化发展,SDN 控制器可以感知全局的网络状态、静态拓扑和动态流表等信息,SDN 交换机只需按照控制器下发的统一规则执行相应的动作.SDN 的这些特点表明在分布式存储网络中采用基于 SDN 的测量方法获取网络状态信息,是一种灵活可行的方式.目前还出现了利用 SDN 技术来优化存储集群的负载的研究.马欢[24]为云计算中心提出了一种分布式 SDN 流表存储架构,使多台 SDN 交换机互相协作实现优化存储空间负载,有效地解决数据中心大流表存储问题.Lai 等人[25]将软件定义网络构架应用在数据存储中心网络路由研究中,提出了基于流分类的软件定义网络路由算法,优化了链路利用率、存储节点负载和分组端到端时延.Girisankar 等人[26]设计了软件定义网络架构,将优化算法集成在光数据网络中心并执行数据存储数据流调度,有效减少了集群负载.Di 等人[27]考虑到现在多种 QoS 流量技术存在的不足,利用 SDN 控制层和数据层相分离的思路,提出了基于 OpenFlow 协议的 QoS 流量方法,提高了存储服务质量的灵活性和可靠性.宋佳[28]采用 SDN 网络架构,提出了基于通信链路带宽均衡的存储节点资源调度策略,有效地优化了图像信息存储的响应时间.山东大学的刘帅[29]提出了基于需求的 SDN 动态云存储服务,通过 SDN 动态调整 SDN 控制器域,减少控制平面到数据平面之间的时延,从而提高存储数据传输效率,为 SDN 的大规模部署和基于 SDN 的云存储的应用提供了很好的理论依据和实际支撑.这些用 SDN 来优化存储系统网络路由和链路方面的研究说明了在分布式存储网络中采用 SDN 技术的可行性.

而目前基于 SDN 提供的网络测量功能按照流量测量框架分为基于 OpenFlow 协议[30]和 OpenFlow 协议之外的流量测量方法.基于 OpenFlow 协议的测量方法中,OpenTM[31]通过获取 OpenFlow 控制器中的路由信息,定时地读取不同的交换机上活跃的流表项统计数据从而构建出全网的流量矩阵.但是,周期性地向交换机发送请求会增加网络开销.FlowSense[32]采取基于流量的被动测量方式,但是会产生过量剩余流量.类似的基于 OpenFlow 协议的流量测量方法还有 OpenNetMon[33]、PayLess[34]等.OpenSketch[35]使用类似 OpenFlow 的软件定义方法,但是 OpenSketch 的实现需要对交换机的部分硬件进行重新设计,代价太大.OpenFlow 协议之外的流量测量方法还有 OpenSample[36]、DevoFlow[37]、PLANCK[38]等,它们的实现难度和复杂性高于基于 OpenFlow 协议的方法.综合比较可以发现 OpenFlow 协议带来的网络开销是最小的.本文正是利用基于 OpenFlow 协议的流量测量方法结合 Ryu 控制器获取 Ceph 集群的网络信息和负载状况.Ryu① 是 Python 语言实现的开源控制器,其丰富的库函数和应用组件有着很高的开发效率并且较好地支持 OpenFlow 协议各个版本.

3 Ceph 数据存储过程及问题描述

本节通过分析 Ceph 系统的数据存储过程介绍本文要解决的问题.Ceph 分布式文件系统能提供的可靠性和可伸缩性离不开底层组件 RADOS[39](Reliable Autonomic Distributed Object Store)的支持.RADOS 包含大量的对象存储设备 OSD(Object Storage Device)和少量的负责管理 OSD 的 Monitor 节点.RADOS 引入了存储池(Pool)的概念,存储池包含用户指定数量的放置组(Placement Group,PG),存储数据时,文件会被拆分成一定大小的对象,每个对象会被唯一映射到一个 PG 中,PG 通过 CRUSH[3]算法映射到一组 OSD 中,整个映射流程可以用图 1 表示.

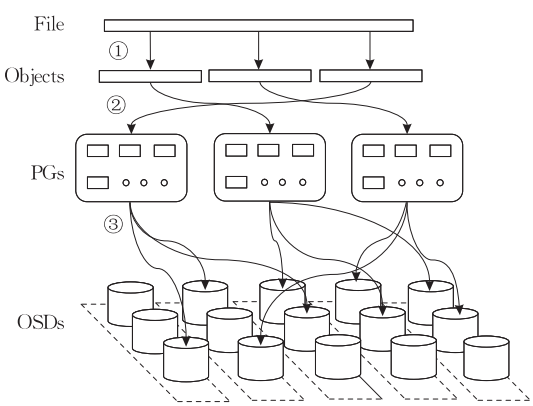

图 1 Ceph 云存储系统数据映射过程

① Ryu.RYU SDN Framework[EB/OL]. https://osrg.github.io/ryu-book/en/html. 2017

完整的数据映射路径中对系统中数据选择 OSD 影响最大的两个步骤是数据对象到 PG 的映射和 PG 到 OSD 的映射. 数据对象到 PG 的映射过程可以用式(1)所示.

$$\mathrm{HASH}(\mathrm{oid})\&mask=pgid \tag{1}$$

式(1)表示将文件划分后得到对象名标识符(oid)作为哈希函数的输入会被构造成随机的值,哈希后的值和"*mask*"值按位进行与操作后生成 PG 的编号 *pgid*. 一般 PG 的数量规定为 2^N,其中 N 为整数,取决于搭建集群的规模,*mask* 的值取为 PG 总数减 1. 这种映射机制可以保证在大量的对象和 PG 情况下数据对象的分布近似均匀,同时保证选择的随机. 为了与其他存储池中的 PG 编号区分,每个 PG 前还要加上存储池编号,PG 的实际编号 *pgid* 是包含 *pool_id* 和 *pg_id* 的结构体变量.

PG 到 OSD 的映射过程是在 CRUSH 算法中实现的,这个过程可以用式(2)描述:

$$\mathrm{CRUSH}(pgid,CRUSH\_map,ruleno)=(OSD_0,OSD_1,OSD_2,\cdots,OSD_i) \tag{2}$$

其中 OSD_i 表示第 i 个对象存储设备,其数量由指定的副本数决定,*pgid* 是第二次映射的输出值,*CRUSH_map* 是包含集群拓扑结构等信息的集群映射(Cluster Map),*ruleno* 是副本放置规则(Placement Rules)的编号,用户可以根据集群状态配置集群映射和副本放置规则. PG 到 OSD 的映射关系并不是固定的,在增删 OSD 节点或 OSD 出现故障时即集群映射发生变化或副本放置规则改变时,PG 到 OSD 的映射结果会动态调整.

CRUSH 算法是把文件对象映射到存储设备中的一种可控的伪随机算法,在 RUSH[40] 系列的算法基础上进行改进得到,可以解决可靠性和数据复制问题,并且提高了性能和灵活性,实现去中心化,避免存储系统由于中心节点带来的瓶颈效应. 通过 CRUSH 算法来选择存储位置是随机的,但存在的问题是:从 CRUSH 算法的函数调用过程可以看出,存储节点选择函数 *crush_do_rule* 的输入参数包括权重向量、副本放置规则、CRUSH Maps 等,其中 CRUSH Maps 记录的是存储集群的层级结构与副本映射以及仅仅由存储容量换算后的正比权重,权重值是不同类型的 Buckets 中选择算法在选择 PG 分布到 OSD 的唯一约束条件. 这其中的存储容量作为决策的唯一因素只能满足集群中的数据空间分布的均匀性,却忽略了底层网络和 OSD 负载对集群性能带来的影响,这样就会造成在某 OSD 节点网络性能很差和负载很高的情况下仍然被选中为主 OSD,负责客户端的读写操作,从而导致集群性能下降.

另外 Ceph 通过副本模式来保证数据的可靠性,Ceph 可以采用三种副本一致性方案:primary-copy,chain 和 splay. 三种方案各有优势,primary-copy 处理读写操作都是在主 OSD,其优势在于以并行的方式更新所有副本,即使副本数增加,往返时延(Round-Trip Time,RTT)总是保持不变,这也是本文中用到的读写方式,其读写过程如图 2 所示,客户端写操作时,数据先写入到主 OSD 后,主 OSD 负责同时向一个或者多个次 OSD 写入副本,只有主次 OSD 都写入完成后,主 OSD 才向客户端返回写入成功. 客户端读操作时只需要向存储数据所在的主 OSD 发送请求并等待数据的返回. 客户端在每次读写操作中选中的主次 OSD 也都需要通过 CRUSH 算法计算得到.

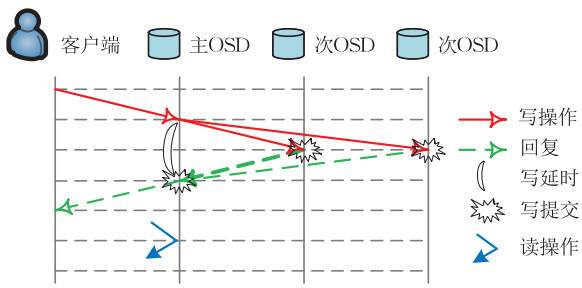

图 2　primary-copy 读写过程

综合以上分析 Ceph 中计算数据分布的 CRUSH 算法和多副本一致性方式,可以发现 Ceph 是将 OSD 权重作为主次 OSD 节点选择的唯一标准,而 OSD 权重又是以剩余存储容量作为唯一决定因素,这意味着剩余存储容量是数据分布和读写操作中选择主次 OSD 的唯一决定因素. 存储容量作为决策因素只能满足集群中的数据空间分布的均匀性,却忽略了底层网络和 OSD 负载对集群性能带来的影响,这样就会造成在某 OSD 节点网络性能很差和负载很高的情况下仍然被选中为主 OSD 负责客户端的读写操作,集群性能会随之下降. 这就有必要综合考虑存储网络的网络状态信息和底层节点负载信息来优化 Ceph 云存储系统模型,建立能提高性能的存储节点选择策略.

4　基于 SDN 构架的 Ceph 分布式存储系统存储节点选择方法

由前一节的介绍可以看出,本文所做工作是以

优化 Ceph 存储的读写性能为目的，建立 Ceph 存储节点选择的一种新方法，其中需要综合考虑网络状态信息和 OSD 主机节点的负载信息，这就首先需要有效地对这两方面的信息进行采集. 为此本文在 Ceph 分布式存储系统中融合了软件定义网络(SDN)的构架，对网络流量等状态信息进行测量. 这样在系统设计中采用软件定义网络和 Ceph 集群结合的解决方案，由于在主次 OSD 选择中依据的是权重因子，而权重因子中综合考虑了节点容量、网络以及负载状况，因此本节先介绍设计的系统整体架构，然后介绍确定网络以及负载状况所用到的参数、多权重因子决策方法及主次 OSD 存储节点选择算法，最后介绍各个模块的具体实现方法.

4.1 系统架构

基于 SDN 技术设计的分布式存储系统架构如图 3 所示，系统架构底层由监控节点和存储节点组成，存储节点和监控节点都是普通的服务器，每个存储节点可以包含多个 OSD，监控节点维护集群所有节点的全局配置信息. OpenFlow 交换机连接所有的服务器，负责数据之间的传送. 系统架构顶层由监测模块利用 SDN 控制器监测需要的 OSD 信息，Ceph 监控节点中的选择模块远程调用 SDN 控制器收集到的信息为选择主次 OSD 做出决策参考. 这种系统架构利用控制平面和数据平面相分离的网络模式方便我们对底层网络和负载情况进行监测. 这样设计的系统构架，可以方便软件定义网络(SDN)采集网络状态信息.

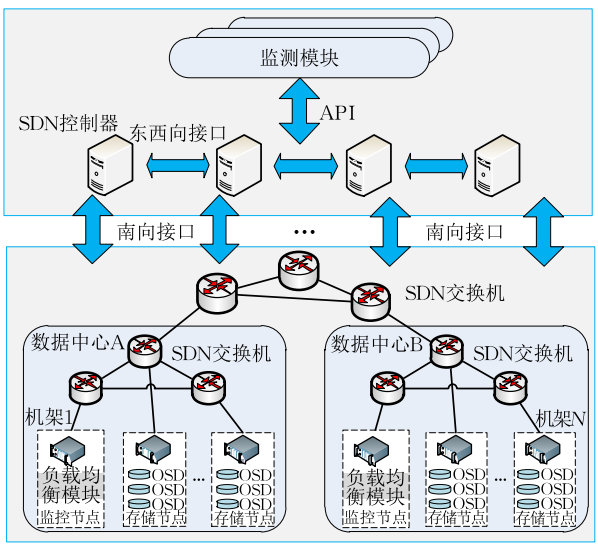

图 3 融合 SDN 技术的分布式存储系统架构

4.2 基于多属性决策模型的 OSD 存储节点选择方法

在以上基于软件定义网络的 Ceph 集群的系统架构的基础上，从本小节开始分别介绍本文设计的 OSD 存储节点选择算法，包括权重因子等参数的确定方法、基于多属性决策的主次 OSD 选择算法. 以及多属性决策方法具体部署的方式和工作流程.

4.2.1 OSD 权重因子的确定

依据第 3 节中介绍的 Ceph 数据存储过程可知，Ceph 选择主次 OSD 存储节点时依据的是 OSD 节点的权重，而这些权重的确定仅仅考虑了 OSD 节点的剩余存储容量. 对此本小节在确定 OSD 权重因子时增加了 OSD 节点的网络状态和 OSD 节点的负载情况的因素，能准确反映 OSD 网络状态和负载情况选取的四个指标及依据如下：

(1) OSD 的剩余带宽 B. 剩余的网络带宽直接影响客户端读写数据的速度.

(2) OSD 的 I/O 负载 L. 磁盘的 I/O 负载反映了该 OSD 正在的读写情况，如果负载很大，OSD 响应客户端的读写数据请求延时就会增加，极端情况下会导致 OSD 节点死亡.

(3) OSD 的 CPU 利用率 C 和内存利用率 R. OSD 是智能的半自治设备，自我管理数据冗余、故障检测和故障恢复需要消耗的 CPU 和内存.

选用这四个指标综合考虑了 OSD 网络状态和负载情况. 考虑到这些指标并不一定存在相关性，特别是在异构网络和异构节点时的情形下，比如带宽和节点处理能力不同，那么各个异构 OSD 节点的剩余带宽与节点的 I/O 负载不会有必然的联系，这是由于剩余带宽不仅与当前业务量有关，还与本身硬件基础设施相关，即使相同的主机负载并不会看出网络剩余带宽的多少. 相关文献也对此有过讨论：文献①用收发文件的实验证明了 I/O 与节点间的传输带宽并无关联，文献[41]证明了内存和 CPU 利用率在某些情况下的相关性但 I/O 负载和 CPU 在某些情况下不相关. 而考虑到对于 Ceph 云存储系统来说，OSD 是智能的半自治设备，自身的状态管理等操作需要耗费一定的 CPU 及内存资源，耗费的资源用于节点的内存、CPU 利用率的管控，因此 OSD 的 CPU 及内存利用率和可用带宽、I/O 负载也不存在必然的相关性，选取的这几个维度可以比较好的反映出 OSD 节点网络状况和负载情况.

4.2.2 基于多属性决策的主次 OSD 选择算法

在确定能准确反映 OSD 网络状态和负载选取

① https://www.violin-systems.com/blog/understanding-io-random-vs-sequential/

情况的四个指标后，下面要做的就是依据各个 OSD 对象的这四个属性值综合选出最优的 OSD 节点，这可以归结于数学统筹学中的多属性决策问题. 多属性决策[42]在系统工程领域有着广泛的理论和应用背景，逼近理想解排序法(Technique for Order Preference by Similarity to Ideal Solution，TOPSIS)是一种有效的多属性决策方案，该方法从归一化的原始数据矩阵构造出决策问题的正理想解和负理想解，计算各方案与正理想解和负理想解的距离，依此作为评价依据. 其主要计算流程为首先对各项指标进行归一化处理，依照其重要性分配权值进行加权处理构造加权规范化矩阵，之后确定正负理想方案，从加权规范化矩阵中选出各项指标的参数值的最大值和最小值，计算与正负理想方案的距离最后求出相对贴近度.

除存储容量外，衡量 OSD 优劣的四个指标中剩余带宽 B 为正指标，值越大 OSD 网络性能越好. I/O 负载 L，CPU 利用率 C 和内存利用率 R 为负指标，值越小 OSD 性能越好，在建立 TOPSIS 模型中负号表示负指标，设计的主次 OSD 选择算法设计如下：

步骤 1. 构造 OSD 权重因子决策矩阵 $\boldsymbol{M}$，通过式(4)归一化处理得到规范化的决策矩阵 $\boldsymbol{M}'$，f_{ij} 是矩阵 $\boldsymbol{M}$ 中的元素，i 和 j 分别是行号和列号，n 为 OSD 的个数.

$$\boldsymbol{M}=\begin{bmatrix} B_1 & -L_1 & -C_1 & -R_1 \\ B_2 & -L_2 & -C_2 & -R_2 \\ \vdots & \vdots & \vdots & \vdots \\ B_n & -L_n & -C_n & -R_n \end{bmatrix} \tag{3}$$

$$M'_{ij}=\frac{f_{ij}}{\sqrt{\sum_{i=1}^{p} f_{ij}^2}},\ i=1,2,\cdots,n;\ j=1,2,3,4 \tag{4}$$

步骤 2. 剩余带宽 B、I/O 负载 L、CPU 利用率 C 和内存利用率 R 影响 OSD 节点性能的比重是不同的，带宽和 I/O 相对比重大. 选取合适的加权系数 W，对式(6)构造规范化的加权决策矩阵 $\boldsymbol{Z}$，式(5)中权重的值一般通过实验的方式获取[43].

$$W=[W_B W_L W_C W_R] \tag{5}$$

$$\boldsymbol{Z}=W_j\times M'_{ij},\ i=1,2,\cdots,n;\ j=1,2,3,4 \tag{6}$$

步骤 3. 确定加权决策矩阵 $\boldsymbol{Z}$ 的正理想解和负理想解

$$Z^+=(Z_1^+,Z_2^+,Z_3^+,Z_4^+)=\max_i\{Z_{ij}\mid j=1,2,3,4\} \tag{7}$$

$$Z^-=(Z_1^-,Z_2^-,Z_3^-,Z_4^-)=\min_i\{Z_{ij}\mid j=1,2,3,4\} \tag{8}$$

步骤 4. 计算每个 OSD 到正理想解和负理想解的距离 D^+ 和 D^-

$$D^+=(D_1^+,D_2^+,\cdots,D_n^+),$$
$$D^-=(D_1^-,D_2^-,\cdots,D_n^-) \tag{9}$$

$$D_i^+=\sqrt{\sum_{j=1}^{4}(Z_{ij}-Z_j^+)^2},$$
$$D_i^-=\sqrt{\sum_{j=1}^{4}(Z_{ij}-Z_j^-)^2} \tag{10}$$

步骤 5. 计算每个 OSD 与最优 OSD 对的相对贴近度 C_i^+，值越大表示该 OSD 性能越好

$$C_i^+=\frac{D_i^-}{D_i^++D_i^-},\ i=1,2,\cdots,n \tag{11}$$

这样在 CRUSH 算法原有的基础上在选取主次 OSD 存储节点时考虑了网络性能和存储节点负载的四个指标，以此作为 OSD 权重因子的约束条件，然后通过建立和求解多属性决策模型就可以确定 OSD 存储节点位置.

4.3　监测模块的设计与实现

要实现以上设计的基于多属性决策的主次 OSD 选择算法，就要在系统中采集到前述的几个反映 OSD 网络状态和节点负载情况的几个指标，以实现图 3 中的监测模块. 而基于 SDN 技术获取这些信息离不开 SDN 控制器对底层 SDN 交换机的获取，因此监测模块的实现首先需要获取 SDN 底层交换机的信息，比如支持的 OpenFlow 协议版本、交换机端口信息等. 在获取了 SDN 交换机信息的基础上，对于网络剩余带宽和 OSD 负载的监测信息的获取可以分别利用主动监测和被动两种方式：通过主动方式使 Ryu 控制器周期地主动下发查询信息，通过交换机回复的信息获取交换机端口能力、端口发送的字节数来获取 OSD 网络信息；通过被动方式使 Ryu 控制器被动接收交换机发送的消息，通过解析消息中附加的记录 OSD 负载信息的数据包实现对 OSD 负载的监控，被动方式不需要控制器周期下发请求消息.

采集数据分为两个阶段，第一个阶段为 SDN 控制器通过主、被动方式获取存储节点的信息. 为了防止 SDN 控制器下发的监测包造成网络拥塞，实验中将主动方式监测的时间间隔设为 1s，被动方式的时间间隔等同于存储节点收集自身信息所耗费的时间(非定值). 第二阶段为 Ceph 监控节点远程采集暂存在 SDN 控制器上的数据. 两个阶段的采集数据过程发送的每个 OpenFlow 监测包只有 32 byte，占用

的带宽达不到总带宽的万分之一，所以对网络来说可以忽略不计. 而且 OpenFlow 交换机、SDN 控制器、Ceph 监控节点都是并行化的同时处理各自事务，SDN 控制器和 Ceph 存储节点周期性发送的 OpenFlow 监测包大小为 32 byte，在发包过程中产生的测量时延为微秒级别，可以忽略其对 SDN 控制器采集的毫秒级别时延带来的影响.

基于以上实现过程的分析，本节首先实现了 SDN 控制器对 SDN 交换机信息的获取，然后再使用 SDN 控制器的主动方式通过 SDN 交换机来实现网络剩余带宽的周期性获取，最后使用 SDN 控制器的被动方式获取底层 OSD 负载状态信息，包括 OSD 底层负载数据包的封装、解析过程以及如何通过 SDN 交换机发送数据包到 SDN 控制器.

4.3.1 SDN 控制器对 SDN 交换机信息的获取

本文在系统中选择 Ryu 作为 SDN 控制器，在 Ryu 控制器上实现监测模块的应用开发，Ryu 是由日本 NTT 公司负责设计研发的一款开源 SDN 控制器. Ryu 中的所有功能、接口都完全由 Python 语言实现，使用者可以用 Python 语言在其上实现自己的应用. 它的作用是通过南向接口指定的协议下发流表发现设备以及感知链路状态并生成网络拓扑结构，实现对网络设备的监测，同时制定相应的策略实现统一的控制功能. 监测模块通过调用 Ryu 中定义的 OpenFlow 协议 API 实现协议消息的监听，构造消息实例和对网络协议数据包进行拆包. Ryu 中 OpenFlow 协议事件类(ofp_event)定义所有的事件，通过在应用中注册处理函数，监听四种 OpenFlow 协议消息：交换机信息(SwitchFeatures)、传入数据包信息(PacketIn)、带宽信息(PortStatsReply)、时间统计信息(PortDescStatsReply)，利用 ofp_parser 类构造所需的消息实例并下发给交换机完成请求，Packet library 实现对网络协议包的解析.

使用 Ryu 控制器与交换机建立连接并获取交换机信息的过程如图 4 所示，交换机初始化与控制器之间的连接，连接方式有 TCP(或 TLS)连接，每一个交换机实体必须发送一个 OPFT_HELLO 报文(报文中设置交换机支持 OpenFlow 协议的最高版本)，握手成功后控制器发送一个 OFPT_FEATURES_REQUEST 报文，交换机在收到报文之后回复 OFPT_FEATURES_REPLY 以及其系列在函数库中定义的报文作为响应.

函数 *switch_features_handler(self,ev)* 用于

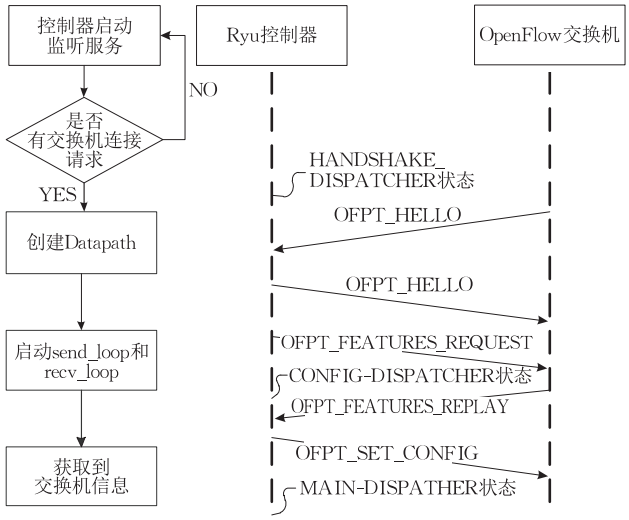

图 4 Ryu 控制器获取交换机信息过程

处理 EventOFPSwitchFeatures 事件，应用中每个注册交换机都会生成 datapath 对象，datapath.id 用于标识不同的交换机，交换机的硬件信息也会作为 SwitchFeatures 应答报文的一部分被发送给控制器，这样完成控制器对交换机信息的获取.

在 OpenFlow 交换机握手协议完成之后，控制器收到 OFPT_FEATURES_REPLY 报文后一般会新增 Table-miss 流表，用于处理 Packet-In 消息，比如无法匹配流表的报文会转发到控制器. 自定义 *add_flow(datapath, priority, match, actions, buffer_id)* 函数，参数指定流表下发的交换机、流表优先级、匹配选项、动作，交换机的缓存号，对于 Table-miss 流表一般指定优先级为 0 即最低优先级，指定 match 的匹配函数为 *parser.OFPMatch()* 表明空 match 可以匹配所有报文，actions 中指定转发的端口为保留端口 Controller 即无法匹配的报文全部转发到控制器. *add_flow()* 函数实现控制器下发流表到交换机是通过 *datapath.send_msg()* 接口函数，构造完流表后调用该函数下发到交换机. 在建立了 SDN 交换机和 SDN 控制器之间的联系之后，就可以进行后面的 SDN 控制器对于带宽的主动下发信息测量以及 SDN 控制器被动接收 OSD 底层节点返回负载的信息包的步骤.

4.3.2 主机剩余带宽监测

剩余带宽作为衡量存储节点之间链路性能的指标之一，更加合理的剩余带宽是取该路径上所有链路剩余带宽的最小值，这可以通过获取各个链路上的各个交换机端口上的信息进行测量. 这里为了方便描述，假设主机直连单个 SDN 交换机的简单网

络的情形，但可以向多个 SDN 交换机互联情形进行拓展.

主机剩余带宽的监测是通过控制器周期下发 OFPPortDescStatsRequest 和 OFPPortStatsRequest 消息到交换机，接收交换机的回复获取端口的带宽能力和端口的接发包字节数. 监测的流程如图 5 所示.

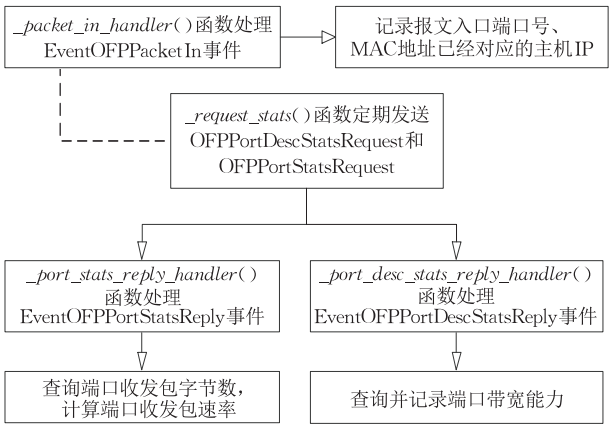

图 5　主机剩余带宽监测流程

初始化的交换机与控制器建立连接后，由于交换机没有路由信息的流表存在，所以报文无法完成正常匹配，Table-miss 流表会指定通过 Packet-In 消息转发到控制器，$\_packet\_in\_handler(self,ev)$ 函数处理 Packet-In 消息事件，ev 是事件的变量名，不同交换机端口发过的数据包入口通过 $ev.msg.datapath.id$，$ev.msg.match['in\_port']$ 获取交换机 id 和端口 id 的元组唯一标识. 每个端口对应的主机 IP 和 MAC 地址需要通过 $get\_protocols()$ 函数解析对应的 Arp 和 Ethernet 网络数据包的包头信息，字典 $ip\_to\_port\{(dpid,in\_port),src\_ip\}$ 记录不同交换机端口下连接的主机 IP，$mac\_to\_port\{dpid:\{src:in\_port\}\}$ 记录端口主机 MAC 地址.

交换机通过 $\_request\_stats$ 函数周期地同时向所有交换机下发查询端口带宽能力和端口收发包状态的消息，在收到交换机回复消息后 $\_port\_desc\_stats\_reply\_handler()$ 执行处理 EventOFPPortDescStatsReply 事件，应用程序中定义用于记录端口属性的字典 $port\_features\{dpid:\{port\_no:(config,state,curr\_speed)\}\}$，其中 $curr\_speed$ 记录了对应主机端口带宽能力. $\_port\_stats\_reply\_handler()$ 函数定义为处理 EventOFPPortStatsReply 事件，在获取发送包字节数 tx_bytes，接收包字节数 rx_bytes 以及查询的流表生存时间的数组 $(duration\_sec,duration\_nsec)$ 后通过 $\_get\_speed()$ 函数计算端口当前速率，计算公式如下：

采集数据间隔时间 T：

$$T_{pre}=(duration\_sec+duration\_nsec/10^9)_{pre},$$
$$T_{cur}=(duration\_sec+duration\_nsec/10^9)_{cur},$$
$$T=|T_{cur}-T_{pre}| \tag{12}$$

其中 pre 与 cur 分别表示前一时间间隔与当前时间间隔.

端口发包速率：

$$tx\_speed=\frac{(tx\_bytes_{T+1}-tx\_bytes_T)\times 8}{T} \tag{13}$$

端口收包速率：

$$rx_{speed}=\frac{(rx\_bytes_{T+1}-rx\_bytes_T)\times 8}{T} \tag{14}$$

端口流量速率：

$$speed=(tx\_speed+rx\_speed) \tag{15}$$

端口剩余带宽：

$$bw=(capacity-speed) \tag{16}$$

SDN 控制器主动下发消息过程代码中函数 $\_get\_free\_bw(capacity,speed)$ 能够获取当前端口带宽容量 $(capacity)$ 和端口流量速率 $(speed)$，带宽容量与流量速率的差值为端口剩余带宽，计算公式如式(16)，通过查找相应在建立通信时记录端口和主机 IP 的映射 $ip\_to\_port\{\}$ 即可获得对应主机网络剩余带宽. Ryu 控制器解析处理 Packet-In 消息事件程序伪代码如表 1.

表 1　Ryu 控制器解析包获取信息伪代码

输入：Packet_in 的 UDP 报文
输出：底层主机收集存储节点的信息

```
@set_ev_cls(ofp_event.EventOFPPacketIn,
            MAIN_DISPATCHER)
Packet_in 消息修饰符
procedure: _packet_in_handler(ev)
  msg←ev.msg
  pkt←packet.Packet(msg.data)
  arp_header←pkt.get_protocols(arp.arp) 地址协议
  ip_header←pkt.get_protocol(ipv4.ipv4) Ipv4 包头协议
  udp_header←pkt.get_protocol(udp.udp) Udp 包头协议
  if (ip_header and udp_header and udp_header.dst_port==
      12345)
    procedure: _parse_udp(msg.data)
end procedure
procedure: _parse_udp(msg.data)
  eth_data←ethernet.ethernet.parser(msg.data)[2]
                                  解析以太网端口
  ip_data←ipv4.ipv4.parser(eth_data)[2] 解析 Ip 地址
  udp_data←udp.udp.parser(ip_data)[2] 解析 Udp 数据包
end procedure
```

4.3.3　OSD 负载状态监测

OSD 负载状态监测是控制器被动的接收交换机转发过来的消息，通过解析消息中的数据包获取 OSD 负载状态. 发送到交换机的报文为了能转发到控

制器则需要报文无法匹配含有指定目的地址路由信息的流表,然后匹配 Table-miss 流表通过 Packet-In 消息转发到控制器,因此,EventOFPPacketIn 事件触发函数在下发带有路由信息的 PacketOut 报文到交换机时,取消指定目的地址的路由信息的流表下发,保证到特定目的地址的报文能转发到控制器,完成对 OSD 负载的监测. 收集 OSD 状态的程序脚本需要运行在存储集群的 OSD 节点上,程序算法伪代码如表 2 所示.

表 2 OSD 负载收集算法伪代码

输入:Ceph 中主机信息
输出:发送 OSD 负载信息到交换机

```
1.  procedure GetOSD(host)
2.    osd←host
3.  end procedure
4.  procedure GetOSDInfo(osd)
5.    osd_info={}
6.    for i in osd do
7.      osd_info[osd_io[i]]←total_kbytes
8.      osd_info[osd_mem[i]]←mem[host[i]]
9.      osd_info[osd_cpu[i]]←cpu[host[i]]
10. end procedure
11. procedure SendData(osd_info)
12.   socket(AF_INET,SOCK_DGRAM).sendto(osd_info,ADDR)
13. end procedure
```

GetOSD()函数的输入参数是所有存储节点主机,获取每个存储节点主机承载的 OSD 并形成一个字典用于记录映射关系,*GetOSDInfo*()函数收集不同 OSD 的 I/O 读写负载以及内存、CPU 负载并记录在字典 *osd_info*{}中. *SendData*()函数通过 UDP 报文发送该数据到交换机,目的地址 *ADDR* 是用户指定的主机 IP.

控制器收到含有 OSD 负载状态信息的 Packet-In 消息后需要通过 *get_protocols*()解析 IPv4 和 UDP 包头协议,判断是否是指定的 *ADDR* 和端口号,匹配成功后 *_parse_udp*(*dpid*, *port*, *msg_data*)函数会解码出数据包内容,根据报文的输入端口和 *dpid* 标识出发送该报文的主机. 由于,之前已经形成了主机和 OSD 的映射关系,所以主机的剩余带宽和负载信息都会被转录成每个 OSD 主机节点的剩余带宽和负载信息.

4.4 主次 OSD 选择模块

本文副本数量设置为三副本,为了同时得到数据的均匀分布和优异的读写性能,在选取主次 OSD 过程中分两个阶段. 第一阶段:为满足集群存储空间的均衡性,Ceph 以存储容量作为权重为 PG 选择一组 OSD. 第二阶段:所有的 PG 选择完 OSD 后,利用 TOPSIS 模型计算得到 OSD 相对贴近度 C_i^+,调整 OSD 的 Primary Affinity 为相对贴近度值,Primary Affinity 的值为[0,1],决定了 OSD 是否能成为主 OSD.

主次 OSD 选择模块运行在 Ceph 的监控节点,在 Ceph 维护的 Cluster Map 中增加了自定义的 OSD_INFO Map,记录集群中所有 OSD 的剩余网络带宽、I/O 负载、CPU 利用率和内存利用率的信息,通过远程调用 SDN 控制器周期地更新 OSD_INFO Map. 在选择主次 OSD 的第二个阶段,根据 OSD_INFO Map 中的信息设置 OSD 的 Primary Affinity,程序算法伪代码如表 3 所示.

表 3 主次 OSD 选择算法伪代码

输入:OSD_INFO Map
输出:确定 OSD 的主次关系

```
1.  osd_addr,osd_value={};
2.  osd_addr=get_osd_addr(); 获取 osd 对应的主机 ip
3.  if osd is 'down' then
4.    remove osd from osd_addr;
5.  end
6.  osd_value[osd]=TOPSIS(OSD_INFO Map);TOPSIS 模型
                          计算 OSD 相对贴近度 C_i^+.
7.  for osd in osd_addr and osd in osd_value do
8.    osd_primary_affinity=osd_value[osd]
9.    update {osd:osd_primary_affinity}
10. end
```

本节介绍了 SDN 监测模块的设计思路和实现方法,首先介绍了模块设计中 SDN 监测用到的几类 OpenFlow 协议消息以及各类消息需要实现的功能,然后根据不同的消息类型确定对 Ryu 控制开发需要调用的协议 API 接口,解析不同的传统网络协议数据包需要用到的解析函数.

最后,详细描述了模块的实现,Ryu 控制器与交换机建立连接,下发查询流表信息并处理回复消息完成主机剩余带宽的主动监测,通过接收 Packet-In 消息解码数据包信息完成 OSD 负载状态的被动测量.

5 实验及结果分析

5.1 实验及测试环境

为了验证本文提出的基于多属性决策模型的 Ceph 存储系统节点选择方法的有效性,本节将在真实的实验平台下验证设计的原型对 Ceph 读写性能的影响. Ceph 集群有 5 台物理机,1 台监控节点,4 台存储节点,每个存储节点有 3 个 OSD,具体硬件配置如表 4 所列,软件环境:Ceph 版本号 0.94.6、Ryu 版本号 4.8、OpenFlow1.3 协议、操作系统 Ubuntu 14.04.1 LTS.

表 4　物理环境硬件配置

用途	数量	配置
Ceph 集群	5	CPU：Intel(R) Xeon(R) E5-2620 v2@2.10GHz 内存：8GB 网卡：1Gb/s
SDN 控制器	1	CPU：Intel(R)Core(TM) i7-4790@3.60GHz 内存：16GB 网卡：1Gb/s
SDN 交换机	2	Alcatel-LucentOS6860E-48

测试是在相同的环境和测试方案下对现有的 CRUSH 算法以及改进后的 CRUSH 算法对系统整体性能的影响进行对比，选取的性能指标及原因如下：

(1) 读写吞吐量. 吞吐量是系统单位时间可以完成的操作数，反应了系统读写性能.

(2) 操作响应时间. 从读写操作初始化到完成经历的时间，响应时间也反应了系统处理读写请求的能力.

测试中向 Ceph 集群写入数据的存储池 Pool 设定副本数量 3，PG 的数量为 512，设置值一般为 $OSDs \times 100/Replicas$ 附近的一个整数，其中 $OSDs$ 是 OSD 数量，$Replicas$ 是副本数，且 PG 的数量必须是 2 的整数幂.

实验环境结构如图 6 所示，COSBench[44] (Cloud Object Storage Benchmark)是云对象存储服务性能测试的基准工具，COSBench 测试集群由三台主机组成，两台负载生成节点负责向 Ceph 集群发送读写请求并收集性能统计，一台 COSBench 控制节点负责接收用户制定的工作量并协调负载生成节点执行工作量，汇总测试结果.

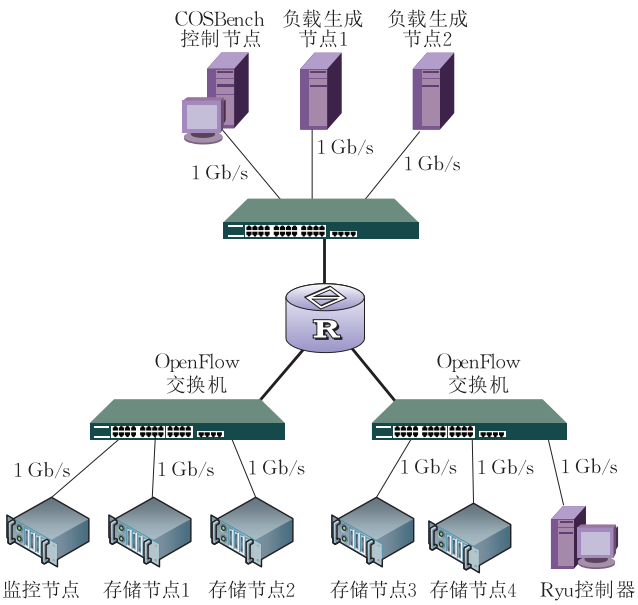

图 6　实验环境结构

为了测试系统对于不同大小对象的读写性能，实验中制定了 4 KB、128 KB、512 KB、1024 KB、4096 KB 五种对象的工作量，比较原有的 Ceph 集群和设计的原型在不同对象下吞吐量和操作从提交到完成的响应时间. 对在 4.2.2 小节中设计的“基于多属性决策的主次 OSD 选择算法”中的加权系数，通过多次实验测试确定出其合适的取值为[0.6，0.3，0.05，0.05].

采集数据分为两个阶段，第一个阶段为 SDN 控制器通过主、被动方式获取存储节点的信息. 为了防止 SDN 控制器下发的监测包造成网络拥塞，实验中将主动方式监测的时间间隔设为 1 s，被动方式的时间间隔等同于存储节点收集自身信息所耗费的时间(非定值). 第二阶段为 Ceph 监控节点远程采集暂存在 SDN 控制器上的数据，通过若干组数据统计得出 SDN 控制器每次采集得到存储节点信息的时间，为了避免 Ceph 监控节点的选择模块连续远程调用到 SDN 控制器收集的数据为同一组数据，再将此时间延长 10%左右作为 Ceph 监控节点远程调用的周期时间.

5.2　结果分析

图 7 和图 8 表示通过实验得到的在不同工作量下的 100%写操作和 100%读操作的吞吐量曲线. 图表中横坐标的工作量以〈*size*〉KB 形式命名，〈*size*〉表示对象的大小，〈*n*〉*c* 表示存放该对象容器的数量，每个容器默认创建 1000 个对象. 设计的原型采用副本强一致性策略，在写操作的过程中，先后在主副 OSD 节点上产生了带宽的消耗，和原有的模型的写操作相比较，总的带宽消耗是相同的，所以设计的原型对于写操作的吞吐量提升并不明显. 设计的

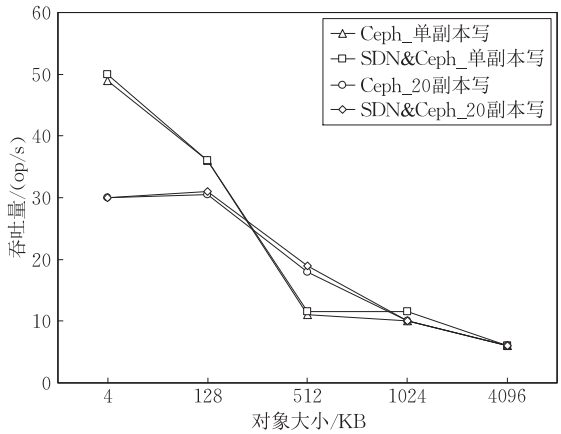

图 7　不同工作量下的 100%写操作吞吐量

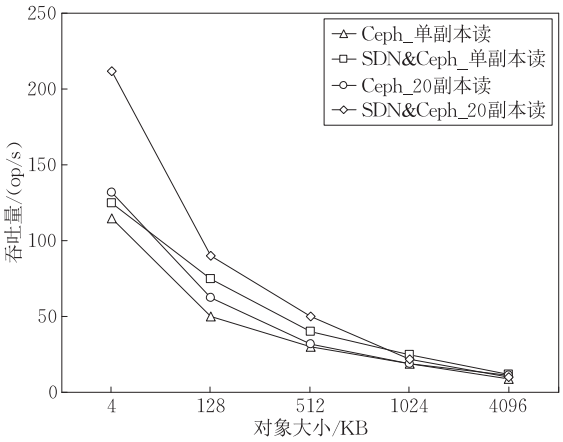

图 8 不同工作量下的 100%读操作吞吐量

原型选取性能较好的节点作为主 OSD,副本模式下读操作只需读主 OSD,所以对读操作的影响很大,对于 1024 KB 以下的小文件的读操作吞吐量有明显提升.

再来观察系统相应时间的改善情况.本文对响应时间的数据使用累积分布函数(CDF)进行统计,图表中的横坐标是响应时间的区间段,左边纵坐标是区间段内完成操作的对象数量用柱状图表示,右边纵坐标是某时间区间在整体中的累积百分比(Cumulative Percentages)用折线图表示,通过第 90 个百分位(90th percentile)所在的时间区间反应出整体的响应时间.图 9 和图 10 分别是工作量为 4 KB_1c 和 4096 KB_1c 的 100%读操作响应时间,设计的原型对于 4KB 对象的 100%读操作响应时间比原有的 Ceph 集群减少了 10 ms 左右,对于 4096 KB 对象的 100%读操作响应时间相对减少了 120 ms 左右.图 11 和图 12 分别是工作量为 4KB_1c 和 4096 KB_1c 的 100%写操作响应时间,如图可见,设计的原型在写操作方面的响应时间并没有得到显著的改善.

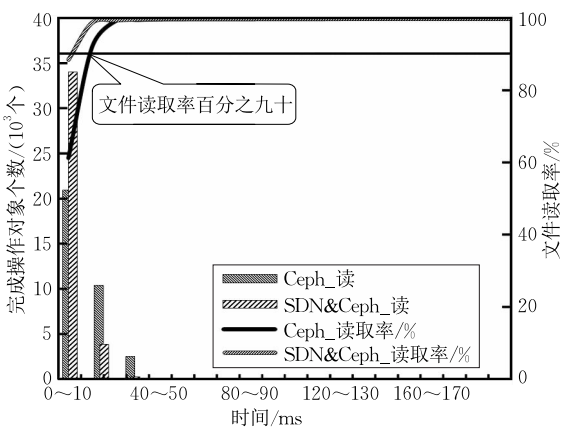

图 9 4KB_1c 的 100%读操作响应时间图

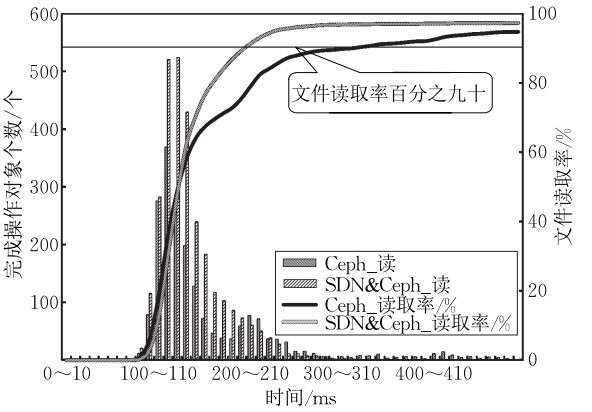

图 10 4096KB_1c 的 100%读操作响应时间

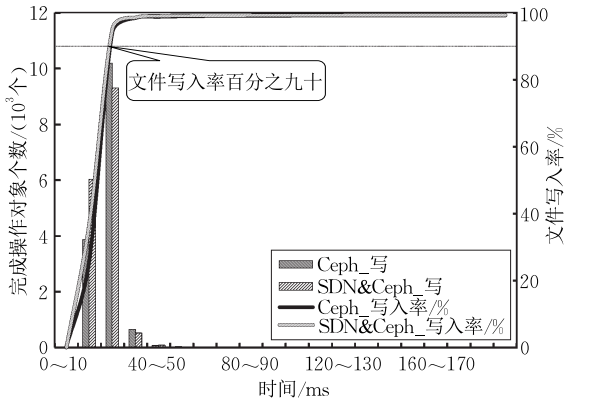

图 11 4KB_1c 的 100%写操作响应时间

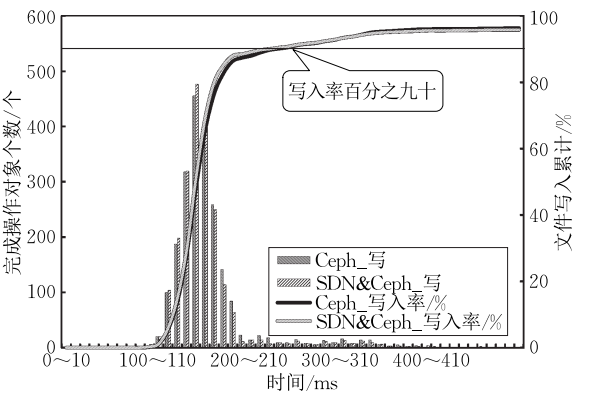

图 12 4096KB_1c 的 100%写操作响应时间

通过上述对不同大小对象的读写性能测试,结果表明设计的原型对于小文件读操作的吞吐量和大文件读操作响应时间有明显地改善,但是写操作性能提升并不明显.这是由于在速率方面,写操作除了在 Object Storage Server 的调度下(其中包括负载均衡计算、分布策略选择等因素)写入数据及其多个副本之外,还需要在元数据服务集群(Metadata Server Cluster)中记录下元数据与日志信息以及与 Object Storage Server 之间的同步操作,具有较大的计算及 I/O 开销,此时带宽的影响不再占据主导

因素；而对读操作而言，读操作通过简单的分布式 hash 变换之后即可映射到对应的 OSD 进行文件 I/O 操作，因此可认为读操作是线性的. 此外，写操作的完成，是以延迟最大的 OSD 节点的所有操作完成为标志，而读操作在我们所设计的方案中，仅需具有最低负载（包括 I/O 负载、CPU 利用率、带宽）的 OSD 节点完成数据传输即可，因此读操作响应时间有较为明显的提高，这也符合系统设计方法的预期. 在吞吐率方面，改进的 CRUSH 算法只是在 PG 选择 OSD 的第二阶段引入了 OSD 的剩余网络带宽和负载状态作为约束条件，PG 在 OSD 上分布仍以存储容量作为权重因子，但是 OSD 上 PG 的主副本数量由第二阶段的权重因子决定，可以根据 SDN 监测到的 OSD 状态实时调整其 PG 主副本数. 副本冗余策略下，读操作在主副本读取数据后返回，写操作需要确定所有的副本 OSD 都写入成功才返回，这样的写入流程导致在写入文件时硬件开销较大，影响了系统的写操作吞吐率，但是由于读取操作的流程计算复杂度较低，因此有着更高的吞吐率. 此外，在实际应用中，读操作的频率要大于写操作，这也是我们工作的实际价值所在.

6　结束语

本文设计了软件定义网络架构下的 Ceph 集群，利用软件定义网络技术感知网络状态，结合节点状态建立起多属性决策的数学模型，通过该模型的求解完成对存储节点的选择. 通过对集群读写性能的测试，与未引入软件定义网络技术的情形相比，实验结果表明设计的存储节点选择模型对于读操作性能提升明显. 由于本文的工作量集中在 Ceph 系统存储节点选择性能优化方面，所以在 SDN 监控的资源消耗问题上还需要做进一步研究，可以继续讨论在测量过程中对交换机的信道带宽资源以及 SDN 控制器的性能的消耗；同时，网络状态是动态变化的，需要细粒度的考虑网络状态的测量和节点选择的决策时刻，以及各个维度权值能动态合理的确定. 同时，在基于 SDN 构建的大规模的存储网络中，单个的 SDN 控制器集中控制和管理会成为瓶颈，这就要对大规模的 SDN 网络使用多个控制器来协同管理，涉及到多个 SDN 控制器的部署、划分及协同通信等研究内容. 这些都是需要后续进行的更有意义的研究工作.

参　考　文　献

[1] Ghemawat S, Gobioff H, Leung ST. The Google file system //Proceedings of the 19th ACM Symposium on Operating Systems Principles. Bolton Landing, USA, 2003: 29-43

[2] Weil S A, Brandt S A, Miller E L, et al. Ceph: A scalable, high-performance distributed file system//Proceedings of the Symposium on Operating Systems Design and Implementation. Seattle, USA, 2006: 307-320

[3] Weil S A, Brandt S A, Miller E L, et al. CRUSH: Controlled, scalable, decentralized placement of replicated data//Proceedings of the 2006 ACM/IEEE Conference on Supercomputing. Tampa, USA, 2006: 13-37

[4] Zhang Y, Debroy S, Calyam P. Network measurement recommendations for performance bottleneck correlation analysis//Proceedings of the 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN). Rome, Italy, 2016: 1-7

[5] Clegg R G, Withall M S, Moore A W, et al. Challenges in the capture and dissemination of measurements from high-speed networks. IET Communications, 2009, 3(6): 957-966

[6] McKeown N. Software-defined networking. INFOCOM Keynote Talk, 2009, 17(2): 30-32

[7] Sun P, Yu M, Freedman M J, et al. HONE: Joint host-network traffic management in software-defined networks. Journal of Network & Systems Management, 2015, 23(2): 374-399

[8] Wang Yi-Jie, Sun Wei-Dong, Zhou Song, et al. Distributed storage key technology in cloud computing environment. Journal of Software, 2012, 23(4): 962-986(in Chinese)
(王意洁，孙伟东，周松等. 云计算环境下的分布存储关键技术. 软件学报，2012，23(4)：962-986)

[9] Huo L, Yi R. Research on metadata management scheme of distributed file system//Proceedings of the 2015 International Conference on Computer Science and Applications (CSA). Wuhan, China, 2015: 37-41

[10] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system//Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). Incline Village, USA, 2010: 1-10

[11] Karger D, Lehman E, Leighton T, et al. Consistent hashing and random trees//Proceedings of the 29th ACM Symposium on Theory of Computing (STOC'97). New York, USA, 1997: 654-663

[12] Burns A J, Lora K D, Martinez E, et al. Building a parallel cloud storage system using OpenStack's swift object store and transformative parallel I/O. Los Alamos National Laboratory, Los Alamos, USA: Technical Report LA-UR-12-23631, 2012

[13] DeCandia G, Hastorun D, Jampani M, et al. Dynamo: Amazon's highly available key-value store. ACM SIGOPS Operating Systems Review, 2007, 41(6): 205-220

[14] Boyer E B, Broomfield M C, Perrotti T A. GlusterFS one storage server to rule them all. Los Alamos National Laboratory, Los Alamos, USA: Technical Report LA-UR-12-23586, 2012

[15] Weil S A. Ceph: Reliable, scalable, and high-performance distributed storage. Annals of Physics (Santa Cruz), 2007: 129-239

[16] Chen Tao, Xiao Nong, Liu Fang. An efficient hierarchical object placement algorithm for object storage systems. Computer Research and Development, 2012, 49(4): 877-899

[17] Huang Qiu-Lan, Wu Jie, Cheng Yao-Dong, et al. Research on data distribution strategy of mass storage system. Computer Engineering and Applications, 2014, 50(10): 1-6(in Chinese)
(黄秋兰, 武杰, 程耀东等. 海量存储系统的数据分布策略研究. 计算机工程与应用, 2014, 50(10): 1-6)

[18] Gudu D, Hardt M, Streit A. Evaluating the performance and scalability of the Ceph distributed storage system//Proceedings of the 2014 IEEE International Conference on Big Data. Washington, USA, 2014: 177-182

[19] Sefraoui O, Aissaoui M, Eleuldj M. OpenStack: Toward an open-source solution for cloud computing. International Journal of Computer Applications, 2012, 55(3): 38-42

[20] Zhang X, Gaddam S, Chronopoulos A T. Ceph distributed file system benchmarks on an openstack cloud//Proceedings of the 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM). Bangalore, India, 2015: 113-120

[21] Han Y, Lee K, Park S. A dynamic message-aware communication scheduler for Ceph storage system//Proceedings of the 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W). Augsburg, Germany, 2016: 60-65

[22] Edwin H M, Liang Y, Jiang W, et al. Optimizing data placement of MapReduce on Ceph-based framework under load-balancing constraint//Proceedings of the 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS). Wuhan, China, 2016: 585-592

[23] Sevilla M A, Watkins N, Maltzahn C, et al. Mantle: A programmable metadata load balancer for the ceph file system //Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Austin, USA, 2015: 1-12

[24] Ma Huan. Research on Key Technology of Network Service Based on SDN in Cloud Environment [Ph. D. dissertation]. Beijing University of Science and Technology, Beijing, 2016 (in Chinese)
(马欢. 云环境下基于 SDN 的网络服务关键技术研究[博士学位论文]. 北京科技大学, 北京, 2016)

[25] Lai Xiang-Wu, Peng Da-Qin, Huang De-Ling, et al. Research on routing algorithm of data center network based on SDN. Wireless Internet Technology, 2016, 1(24): 38-40 (in Chinese)
(赖香武, 彭大芹, 黄德玲等. 基于 SDN 的数据中心网络路由算法研究. 无线互联科技, 2016, 1(24): 38-40)

[26] Girisankar S T, Truong-Huu T, Gurusamy M. SDN-based dynamic flow scheduling in optical data centers//Proceedings of the 2017 9th International Conference on Communication Systems and Networks (COMSNETS). Bangalore, India, 2017: 190-197

[27] Di J, Ma Q. Design and implementation of SDN-base QoS traffic control method for electric power data center network //Proceedings of the 2016 2nd IEEE International Conference on Computer and Communications (ICCC). Chengdu, China, 2017: 2669-2672

[28] Song Jia. Research on the Integrated Storage Network Technology Based on SDN [M. S. dissertation]. Beijing Institute of Technology, Beijing, 2016(in Chinese)
(宋佳. 基于 SDN 的天地一体化存储网络技术研究[硕士学位论文]. 北京理工大学, 北京, 2016)

[29] Liu Shuai. The Control Deployment and Cloud Storage Allocation Problem of SDN Network [M. S. dissertation]. Shandong University, Jian, 2016(in Chinese)
(刘帅. SDN 网络的控制器部署和云存储分配问题研究[硕士学位论文]. 山东大学, 济南, 2016)

[30] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74

[31] Tootoonchian A, Ghobadi M, Ganjali Y. OpenTM: Traffic matrix estimator for OpenFlow networks//Proceedings of the International Conference on Passive and Active Measurement 2010 (PAM 2010). Zurich, Switzerland, 2010: 201-210

[32] Yu C, Lumezanu C, Zhang Y, et al. FlowSense: Monitoring network utilization with zero measurement cost//Proceedings of the International Conference on Passive and Active Measurement 2013 (PAM 2013). Hong Kong, China, 2013: 31-41

[33] Van Adrichem N L M, Doerr C, Kuipers F A. OpenNetMon: Network monitoring in OpenFlow software-defined networks// Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS). Krakow, Poland, 2014: 1-8

[34] Chowdhury S R, Bari M F, Ahmed R, et al. PayLess: A low cost network monitoring framework for software defined networks//Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS). Krakow, Poland, 2014: 1-9

[35] Yu M, Jose L, Miao R. Software defined traffic measurement with OpenSketch//Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI). Berkeley, USA, 2013: 29-42

[36] Suh J, Kwon T T, Dixon C, et al. OpenSample: A low-latency, sampling-based measurement platform for commodity SDN// Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems. Madrid, Spain, 2014: 228-237

[37] Curtis A R, Mogul J C, Tourrilhes J, et al. DevoFlow: Scaling flow management for high-performance networks. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 254-265

[38] Rasley J, Stephens B, Dixon C, et al. Planck: Millisecond-scale monitoring and control for commodity networks. ACM SIGCOMM Computer Communication Review, 2015, 44(4): 407-418

[39] Weil S A, Leung A W, Brandt S A, et al. RADOS: A scalable, reliable storage service for petabyte-scale storage clusters// Proceedings of the 2nd International Workshop on Petascale Data Storage (PDSW'07). Reno, USA, 2007: 35-44

[40] Honicky R J, Miller E L. Replication under scalable hashing: A family of algorithms for scalable decentralized data distribution //Proceedings of the 18th International Parallel and Distributed Processing Symposium. Santa Fe, USA, 2004: 96

[41] Watkins L A. Using network traffic to infer CPU and memory utilization for cluster grid computing applications. ACM Transactions on Embedded Computing Systems (TECS)-Special Issue on Embedded Platforms for Crypto and Regular Papers, 2010, 14(5): 91-97

[42] Chen S J, Hwang C L. Fuzzy multiple attribute decision making methods//Proceedings of the Fuzzy Multiple Attribute Decision Making Conference on Springer. Berlin, Germany, 1992: 289-486

[43] Luo Y Q, Xia J B, Chen T P. Comparison of objective weight determination methods in network performance evaluation. Computer Application, 2009, 29(10): 2624-2626

[44] Zheng Q, Chen H, Wang Y, et al. COSBench: A benchmark tool for cloud object storage services//Proceedings of the 2012 IEEE 5th International Conference on Cloud Computing. Honolulu, USA, 2012: 998-999

WANG Yong, born in 1964, Ph. D., professor, Ph. D. supervisor. His main research interests include cloud computing, network traffic classification, Information security.

YE Miao, born in 1977, Ph. D., professor, M. S. supervisor. His research interests include network computing, wireless sensor network, pattern recognition and image processing.

HE Qian, born in 1979, Ph. D., professor, Ph. D. supervisor. His main research interests include distributed computing, software defined network.

HUAN Yi-Ming, born in 1993, M. S. candidate. His main research interests include cloud computing, distribute storage.

KANG Wen-Jie, born in 1992, M. S. candiate. His main research interests include cloud computing, distribute storage.

Background

The traditional storage model cannot cope with massive data storage capacity scalability, data reliability and high performance, cloud storage systems came into being under this background. Cloud storage systems use distributed file systems and other technologies to assemble different storage devices into a pool of resources through network connections, unified provision of storage services, with high scalability, high reliability and so on. There are many kinds of distributed file systems for cloud storage, Ceph distributed file system, with its open source nature, and providing uniform storage capability, has been widely concerned in enterprises and scientific research fields.

Data distribution strategy is a key technology in distributed file system, which determines location of data storage, load balancing and fault tolerance of the system. CRUSH algorithm is a pseudo-random data distribution algorithm in Ceph distributed file system, which can distribute data objects and their replicas efficiently in heterogeneous large-scale hierarchical structured storage clusters. However, CRUSH algorithm is deficient in the design of storage node load balance and does not take into account underlying network conditions and node load. When network condition of storage node deteriorates, such as data migration within the cluster takes up a large amount of network bandwidth. At present, the system selects location of data storage only to take storage capacity of nodes as weighting factor, which makes performance of the cluster fall significantly, and reduces the read and write performance of the system. Therefore, it is

important to add network state and node load into CRUSH algorithm to improve load balance. The traditional network architecture, getting network state requires cumbersome configuration and a lot of measurement overhead, Software defined network (SDN) as a new network architecture using control plane and data plane phase separation, through the centralized control plane, SDN Simplifies network measurement and management and provides a flexible and efficient maintenance strategy, so the paper uses SDN technology to complete the monitoring of network and node load. The main innovative work of the paper is summarized as follows: (1) Aiming at the load unbalanced problem of storage node caused by the storage capacity as the constraint condition in CRUSH algorithm, an improved CRUSH algorithm is proposed to add the factors of network state and load in node weight factor, and the determination of weight factor has a finer granularity. (2) In the process of obtaining the network state and load status of the node, the architecture of the SDN and the Ceph distributed file system is designed. Through the development of the SDN controller, the measurement of the node network status and load state are respectively measured by the combination of active and passive methods.

This work is partly supported by the National Natural Science Foundation of China (Nos. 61662018, 61661015, 61831013), Project Funded by the China Postdoctoral Science Foundation (No. 2016M602922XB), Guangxi Innovation-Driven Development Project (Major Science and Technology Project No. AA18118031), Guangxi Innovation-Driven Development Project (Major Science and Technology Project No. AA18118031), Foundation of Guilin University of Technology (No. GUTQDJJ20172000019).