第 43 卷 第 4 期 计 算 机 工 程 2017 年 4 月
Vol. 43 No. 4 Computer Engineering April 2017

·体系结构与软件技术· 文章编号：1000-3428(2017)04-0067-06 文献标志码：A 中图分类号：N945

基于 Ceph 文件系统的元数据缓存备份

詹 玲[1]，方协云[2]，李大平[2]，万继光[2]

（1. 文华学院 信息学部，武汉 430074；2. 华中科技大学 武汉光电国家实验室，武汉 430074）

摘 要： 针对 Ceph 文件系统元数据写操作响应时间较长的问题，提出一种对元数据缓存进行备份的方案。该方案采用多个元数据服务器之间互相备份写缓存数据的方法，保证元数据的可靠性，并设计基于元数据热度的多队列缓存组织结构，以及相应的缓存预取与倒盘算法。实验结果表明，与原始的 Ceph 系统相比，该方案的元数据访问性能提升 15.7% 。

关键词： Ceph 文件系统；元数据；热数据；缓存；备份

中文引用格式： 詹 玲，方协云，李大平，等. 基于 Ceph 文件系统的元数据缓存备份[J]. 计算机工程，2017，43(4)：67-72，83.

英文引用格式： Zhan Ling，Fang Xieyun，Li Daping，et al. Metadata Cache Backup Based on Ceph File System[J]. Computer Engineering，2017，43(4)：67-72，83.

Metadata Cache Backup Based on Ceph File System

ZHAN Ling[1]，FANG Xieyun[2]，LI Daping[2]，WAN Jiguang[2]

（1. College of Information，Wenhua College，Wuhan 430074，China；
2. Wuhan National Laboratory for Optoelectronics，Huazhong University of Science and Technology，Wuhan 430074，China）

【Abstract】 In order to solve the problem of long response time of metadata writing in Ceph file system，a metadata cache backup scheme is proposed. The scheme backups the metadata among multiple servers to guarantee the reliability of the metadata，and multi-queue cache organization structure based on the heat of the metadata and the relevant cache prefetching and destage algorithms are designed. Compared with the performance of the Ceph metadata management subsystem，experiment results show that the performance of the proposed scheme is improved by 15.7% .

【Key words】 Ceph file system；metadata；hot data；cache；backup

DOI： 10.3969/j.issn.1000-3428.2017.04.011

0 概述

经过 20 多年的发展历史，业界已经推出了许多成熟的分布式文件系统，这些系统大致可以分为2 类：学术研究类和商用类。学术研究类的分布式文件系统包含 Coda[1]0，TidyFS[2]，OceanStor[3]，CEPH[4]等，商用的分布式文件系统包含普通并行文件系统（General Parallel File System，GPFS）[5]、谷歌文件系统（Google File System，GFS）[6]、Hadoop 分布式文件系统（Hadoop Distributed File System，HDFS）[7]等。这每一种分布式文件系统针对的应用场景都有所区别，在能够满足绝大部分需求之外，针对某些特殊需求，它们都能够发挥各自的最大优势。

在文件系统中，元数据请求量占总请求量的比例达到 50% 以上，如何提高元数据访问性能成为了文件系统的研究热点。通过缓存元数据来提高元数据访问性能是长期以来的研究热点，如在文件系统中使用非易失性存储器（Nonvolatile Memory，NVM）来缓存元数据，从而提高系统的性能和稳定性等。使用 NVM 来提升整个系统的性能，已经有了许多成熟的研究方案。

文献[8]提出了一种使用 MRAM 来存储元数据的方案 HeRMES，该方案是最早提出的使用磁性随机存储器（Magnetic Random Access Memory，MRAM）来存储元数据的论文之一。文献[9]提出一款为按字节存取的非易失随机存储储存器（Non Volatile Random Access Memory，NVRAM）的文件系

基金项目： 国家自然科学基金面上项目“混合式瓦记录磁盘阵列系统理论、结构和实现技术研究”（61472152）；湖北省高等学校优秀中青年科技创新团队计划项目（T201431）；湖北省教育厅科学技术研究计划指导性项目（B2015189）；文华学院博士基金“高性价比云存储网点的结构和原理研究”（2016Y01）；田家炳基金（1602-07-411）。

作者简介： 詹 玲（1973—），女，讲师、博士，主研方向为网络存储系统；方协云，硕士研究生；李大平，博士研究生；万继光，副教授、博士。

收稿日期： 2016-04-11 **修回日期：** 2016-05-25 **E-mail：** zling_2016@163.com

统 MRAMFS,该系统将数据压缩的功能应用于 inodes,支持最基础的文件数据压缩功能,并且该系统中的数据结构专为随机存储器而设计。文献[10]提出一个为 Flash 设备设计的文件系统 MiNVFS,通过将元数据存储在 NVRAM 和数据存储在 Flash 的方式,从而能够提高该系统的性能。文献[11]设计了一种使用 NVRAM 作为元数据写缓存的文件系统 NVFAT,它使用 NVRAM 作为缓存,不仅提高了文件系统的性能,还提高了文件系统的可靠性。

文献[12]提出了一种混合文件系统 hybridFS,hybridFS 充分利用了硬盘驱动器(Hard Disk Drive,HDD)和固态硬盘(Solid State Disk,SSD)的优点,用一种高性价比的方式,构建一个大规模的虚拟地址空间,HybridFS 利用 SSD 的高性能,同时提供一种灵活的数据结构来有效利用现有文件系统高效率的写性能。文献[13]提出一种新的文件系统 Conquest,该系统使用带有备用电池的缓存作为数据的持久储存设备,它提供 2 种数据储存设备在分布式文件系统中。元数据缓存技术是一个研究热点,目前,已有许多针对分布式文件系统中元数据的研究方案,如刘扬宽提出了一种基于非易失性存储器相变存储器(Phase Change Material,PCM)分布式文件系统的元数据管理机制原型系统[14],文献[15]提出的基于非易失性存储器新型缓存子系统。

前文提到了很多通过高性能存储介质来提高元数据访问性能的方法,然而都需要装备相应的硬件,给实际应用带来了诸多不便。为此,本文提出一种利用系统自带内存来缓存元数据以提高其访问性能的方法,并且通过对元数据进行备份来保证其可靠性。

1 问题根源

在 Ceph 系统中,元数据信息存储在对象存储设备(Object Storage Device,OSD)集群中,元数据服务器(Metadata Server,MDS)集群只用来管理元数据信息,并缓存部分元数据信息,为了保证元数据的一致性与安全性,MDS 中的脏数据需要实时更新到 OSD 中,所以,在元数据请求访问操作时,除了需要访问 MDS 外,还需要访问 OSD。在 Ceph 系统中,元数据的修改流程如图 1 所示。

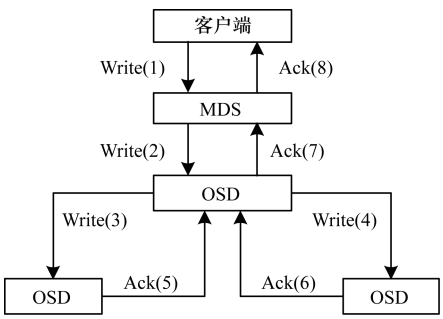

图 1 元数据修改流程

如图 1 所示,当某个客户端需要修改元数据时,需要先找到管理该元数据信息的 MDS 节点,通过该 MDS 节点,访问存储该元数据的 OSD 节点,由于在分布式环境中,单个节点是不可靠的,因此为了保证数据的正确性,数据存储采用三备份策略,因而在修改元数据时,需要在存储该元数据信息的 3 个数据服务器节点中修改相应的元数据信息。在 OSD 集群中,元数据信息更新过程为:先在主 OSD 节点中更新,然后,将主 OSD 节点中更新信息同时拷贝到其他 2 个 OSD 节点中。因而,在更新元数据信息时,需要等待 3 副本修改完成之后,客户端元数据访问请求才正常完成,所以,这种元数据更新管理方案的元数据响应延时比较大。

为了降低元数据的响应延时,可以使用动态随机存取存储器(Dynamic Random Access Memory,DRAM)缓存来减少磁盘访问所需要的时间,根据缓存的服务器类型不同,缓存位置有 2 种:1)元数据服务器节点;2)数据服务器节点。

在数据服务器节点上缓存元数据信息时,相比于 Ceph 系统中原元数据管理子系统,只能够减少缓存命中的热点元数据的磁盘访问延时,没有减少网络传输开销,而且,对于数据服务器节点来说,数据和元数据都是统一管理的,单独对元数据进行缓存管理,增加了对象存储节点中缓存管理的开销。

所以,为了减少元数据的访问延时,本文采取了在 MDS 节点上缓存元数据的方案,本文方案相比于 Ceph 系统中自带的元数据管理子系统,不仅能够减少缓存命中的热点元数据的磁盘访问延时,还能够减少网络传输开销。而且,MDS 节点的功能就是用来管理元数据信息的,所以,使用 MDS 节点缓存元数据信息,能够最大可能地降低元数据访问延时,而且,元数据服务器节点增加的缓存维护开销也相对较小。

另外,鉴于在分布式环境下,组成整个文件系统的每一个节点是不可靠的,本文对 MDS 中的缓存数据进行备份管理,将每个 MDS 上缓存的信息备份到其他 MDS 上,这样就能够避免单个元数据节点失效时,更新的信息被丢失的风险。

2 元数据缓存备份

2.1 系统架构

在 Ceph 系统中,在不影响系统安全性的前提条件下,为了减少元数据的访问延时,需要将 MDS 节点中缓存的元数据信息备份到其他 MDS 中,即当 MDS 节点中缓存的元数据信息发生修改时,需要将该修改的元数据信息同时更新到备份节点中的缓存中,从而,避免单 MDS 节点失效的问题。MDS 节点中缓存的是存储在 OSD 集群中的元数据信息,所以,元数据服务集群和数据服务集群之间就存在元

数据存取交互过程。基于 Ceph 系统的元数据缓存备份方案系统架构如图 2 所示。

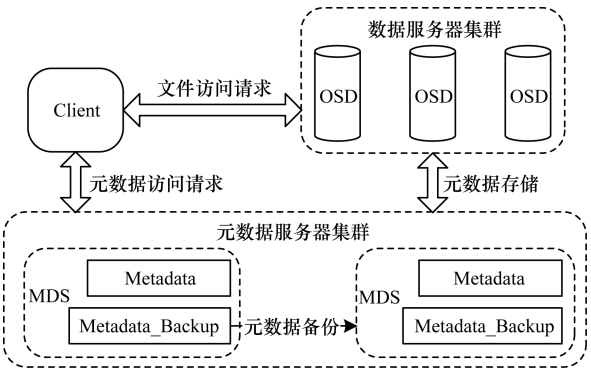

图 2　元数据缓存备份架构

2.2　元数据流

基于 Ceph 系统的元数据缓存备份方案中,为了防止元数据服务节点单节点失效,导致元数据服务节点中缓存的元数据信息丢失,将缓存在 MDS 节点中的元数据信息备份到其他 MDS 节点中。基于 Ceph 系统的元数据缓存备份方案元数据流如图 3 所示。

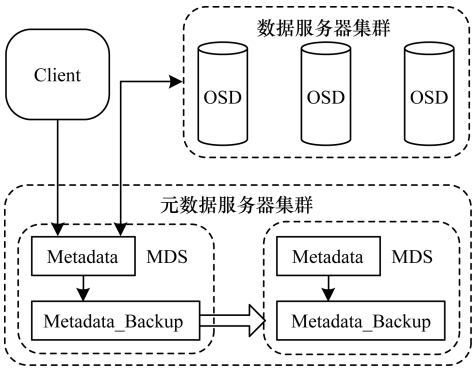

图 3　元数据缓存备份方案元数据流

当元数据服务节点收到客户端的元数据访问请求操作时,先查询 MDS 缓存中是有所需要的信息,如果没有缓存该元数据信息,则需要先将所要访问的元数据信息从数据服务集群中读取上来缓存在 DRAM 中,然后完成该元数据的请求访问操作,最后将更新之后的元数据信息备份到备份目的节点缓存中。

2.3　冷热元数据识别

热数据是指在一段时间内访问频率比较高的数据,冷数据就是指在一段时间内访问比较少或者没有访问的数据。随着时间的流逝,热数据会慢慢变冷,成为冷数据,冷数据也会慢慢变为热数据,所以,在识别冷热数据时,既要考虑数据的访问次数,也要考虑热数据被访问之后的热度值衰减问题。

在文件系统中,元数据信息可以分为目录元数据信息和文件元数据信息。对于目录来说,只要目录下的文件或者子目录被访问,该目录元数据信息就会被访问,所以,相对于文件元数据信息来说,目录元数据信息被访问的机率大,因而,在考虑元数据热度值衰减时,目录元数据信息的热度值衰减速率要慢于文件元数据信息的热度值衰减速率。而且,包含的子目录和文件数目越多的目录,该目录被访问的机率越大,所以,目录包含的子目录和文件数目越多,其元数据信息的热度值衰减速率越小。文件和不同目录的元数据信息热度值衰减曲线如图 4 所示。

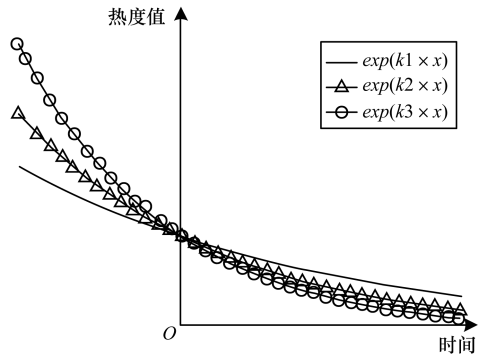

图 4　元数据热度值衰减曲线

在这 3 个元数据信息热度值衰减曲线图中,$exp(k3 \times x)$曲线衰减速率最快,表示文件元数据信息热度值衰减情况;曲线 $exp(k1 \times x)$和曲线 $exp(k2 \times x)$都表示目录的元数据信息的热度值衰减变化情况,其中,$exp(k1 \times x)$曲线表示包含子目录和文件数目多的目录的元数据信息热度值衰减情况,$exp(k2 \times x)$曲线表示包含子目录和文件数目少的目录的元数据信息热度值衰减情况。

按照文件系统中元数据信息访问的基本规律,假设在某一时刻,某元数据信息的热度值为 val,则经过时间间隔 $t(s)$之后,热度值有一定的衰减,此时,对于文件而言,文件元数据信息的当前热度值 $newval$ 可以表示为:

$$newval = val \cdot e^{\ln 0.995 \times t} \tag{1}$$

对于目录而言,假设该目录下包含的文件和子目录数量为 N,则目录元数据信息的当前热度值 $newval$ 可以表示为:

$$newval = val \cdot e^{\frac{\ln 0.99}{N} \times t} \tag{2}$$

基于 Ceph 系统的元数据缓存备份方案中,计算热度值时,在不考虑热度值的衰减条件下,可以通过访问次数来表示元数据信息的热度值,所以,在考虑热度值衰减之后,只需要在元数据信息再次被访问时,重新计算热度值即可。假设元数据信息上一次被访问时的热度值为 val,经过时间间隔 t,元数据信息再次被访问,则此时的热度值 $newval$ 表达式为:

$$newval = \begin{cases} val \cdot e^{\ln 0.995 \times t} + 1 & ,\text{文件元数据信息热度值} \\ val \cdot e^{\frac{\ln 0.99}{N} \times t} + 1 & ,\text{目录元数据信息热度值} \end{cases} \tag{3}$$

2.4　元数据缓存算法

为了降低 Ceph 系统元数据的访问延时,在 MDS 节点中缓存元数据信息,由于每条元数据信息只有几

十字节到几百字节,而每个 MDS 节点中 DRAM 容量非常大,因此在 MDS 中能够缓存的元数据信息条目非常多。所以,在基于 Ceph 系统的元数据缓存备份方案中,采用多级活跃队列来管理缓存的元数据信息,这样能够在命中率比较高的条件下,复杂度也比较低。根据前一节元数据信息的热度值将元数据信息分为不同的活跃级别,每一个活跃级别对应一个队列 Q,元数据缓存替换算法如图 5 所示。

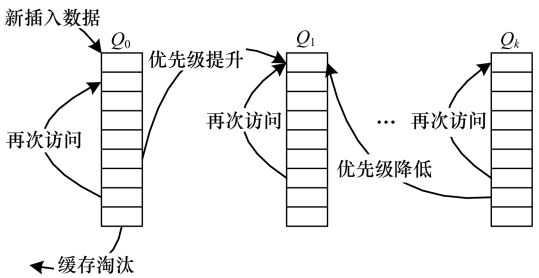

图 5 缓存备份方案的缓存替换算法

如图 5 所示,基于 Ceph 系统的元数据缓存备份方案的缓存算法中,队列 Q_0,Q_1,…,Q_k 分别表示优先级不同队列,而这些队列内部是按照 LRU 管理元数据信息条目的,所以,根据图 5 知,基于 Ceph 系统的元数据缓存备份方案的缓存算法的多级活跃队列的基本操作原则为:

1)新缓存的元数据条目放入队列 Q_0 的头部。

2)每条活跃队列基于 LRU 原则管理缓存条目。

3)在某个队列内的元数据信息热度值上升到该队列的热度值上限,需要提升优先级时,就将该元数据信息条目从当前队列迁移到高一级队列的头部。

4)在某个队列内的元数据信息热度值降低到该队列的热度值下限,需要降低优先级时,就将该元数据信息条目从当前队列迁移到低一级队列的头部。

5)缓存空间不够,需要淘汰数据时,在队列 Q_0 基于 LRU 算法淘汰元数据信息条目。

6)基于 Ceph 系统的元数据缓存备份方案的多级活跃队列中,每一个队列代表不同区间的热度值。在分布式环境下,在一段时间内,会有部分元数据的信息热度值会非常高,如果将热度值均分,会使得队列数量比较大,而且中间部分队列的数据量非常少,所以,采用指数幂的形式划分区间,第一条队列 Q_0 热度值范围为(0,2),其他队列 Q_i 热度值范围为 $[2^i,2^{i+1})$。

为了防止热度值高的缓存元数据信息永远不会被淘汰,需要将这些高优先级的元数据条目进行降级处理,通过定时老化机制,除了第一条队列外,将其他队列上的所有节点都迁移到低一级的队列上,对于热点数据来说,当再次访问时,其优先级还可再次提升,所以,采用定时统一降级方案是可行的,定时老化机制如图 6 所示。

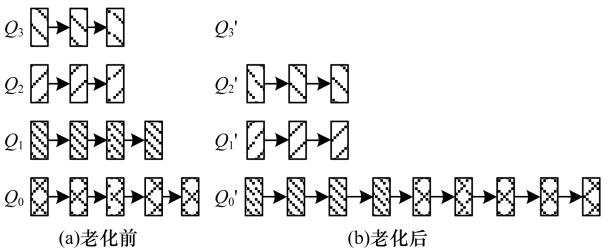

图 6 缓存备份方案的定时老化机制

如图 6 所示,定时老化过程是将高一级链表中的所有节点链接到低一级链表的表头,并且保持原来链表的顺序,在第一级链表上,将第二级链表的所有节点在第一级链表的前半部分,而第一级链表的原来节点都在链表的后半部分,并且保持节点的原来顺序不变化。

2.5 元数据预取

数据预取是为了在处理器访问该数据进行处理之前,提前将数据从慢速存储设备中加载到缓存中,以降低处理器访问数据的延时。在 Ceph 系统中,元数据访问请求是由 MDS 节点处理的,而这些信息存储在 OSD 集群中,所以,在元数据缓存备份方案中,为了降低元数据的访问延时,将元数据信息预取到数据服务节点的缓存中是非常必要的,它能够有效降低元数据的访问延时。

在文件系统中,单个元数据信息包含的数据量一般都比较小,而且,元数据访问操作遵循一些基本规律:

1)对于元数据信息进行访问时,需要冗长的定位处理过程。

2)目录元数据信息平均被访问的概率比文件元数据信息被访问的概率大。

3)元数据信息的访问请求中,读操作比修改操作比例大。

4)根据文件系统中元数据访问操作的基本特性,基于 Ceph 系统的元数据缓存备份方案对元数据的预取操作包含 2 种情况:

(1)从 OSD 集群获取 dentry 信息时,同时预取相应的 inode 信息。

(2)从 OSD 集群获取文件元数据信息时,同时预取同一层的所有文件和目录的元数据信息。

2.6 元数据缓存倒盘

在 Ceph 系统中,元数据信息存储在数据服务器集群中,所以,缓存在元数据服务节点中的元数据信息进行倒盘处理操作时,是为了将缓存中的元数据信息倒入到 OSD 集群中,从而腾出缓存空间用来缓存新的元数据信息。根据倒盘时机的不同,倒盘可以分为 2 类:1)主动倒盘;2)被动倒盘。这 2 种算法在系统中都有使用,只不过主动倒盘是在空间还足够的时候,充分利用系统的空闲时间对数据进行倒盘,而被

动倒盘则是因为存储空间不足而引发的倒盘。

1)被动倒盘

缓存新的元数据信息时,如果缓存容量空闲空间低于阈值(1%)时,就启动倒盘,将热度值最小的部分元数据信息写入数据服务集群中,释放 DRAM 缓存空间,直到缓存空闲空间高于阈值(2%)。

2)主动倒盘

在系统不忙或者空闲时,主动将 DRAM 缓存中的元数据信息写入到 OSD 集群中,此时,不需要删除缓存中的元数据信息,释放缓存空间,因为缓存容量空间还足够,且这些元数据信息还可能再次被访问到。

3　实验测试与结果

3.1　原型实现

基于 Ceph 系统的元数据缓存备份架构中的数据流如图 7 所示。

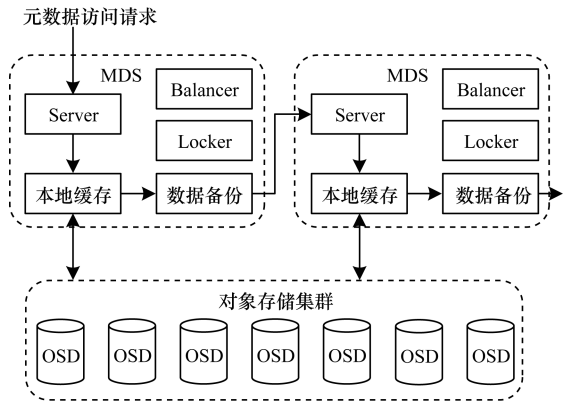

图 7　元数据缓存备份的模块

如图 7 所示,当整个 Ceph 系统中有元数据访问请求时,元数据服务器集群中的某个 MDS 节点会收到该请求,即该 MDS 节点的 Server 模块会收到请求信息,Server 模块收到元数据请求处理消息之后,会在本地缓存模块中进行相应的处理,将元数据缓存在本地中,然后经过数据备份模块,给备份该数据的目的 MDS 节点发送一个备份数据的消息,从而目的 MDS 节点会对元数据进行一个备份处理,将该数据存储在目的 MDS 节点的 DRAM 中,这样,每个更新的元数据信息都会有 2 个副本,不会因某一个 MDS 节点失效而导致更新的数据丢失。

3.2　性能测试平台

实验测试环境 Ceph 系统部署在 6 台服务器上,其中包括 3 台数据服务器(OSD)节点,2 台元数据服务器(MDS)节点,以及一台监控服务器(Monitor Server,MON)节点。6 台服务器连接在局域网中同一个交换机上,相互之间通信,完成消息通信和数据传输。实验测试平台同时使用 MON 节点作为客户端,即在监控服务器节点上挂载 Ceph 文件系统,对整个 Ceph 文件系统的性能测试就是测试该挂载的文件系统性能。各个服务器节点的配置大致相同:华硕主板;Intel(R) Xeon(R) E5606 2.13 GHz CPU;16 GB DDR2 RAM;500 GB 的系统磁盘以及 1 TB的数据磁盘;千兆网卡;操作系统分别为 Ubuntu 14.04 x86_64;内核版本为 Linux 3.13.0-24。

filebench 是一个标准文件系统的 benchmark 工具,能够自动化测试文件系统的性能,对于文件系统的性能测试非常合适。使用 filebench 测试工具测试设计方案的性能时,应用了 fileserver(文件服务)、makedirs(目录生成)、listdirs(目录罗列)和 createfiles(创建文件)4 种负载,这 4 种负载线程数目均设置为 8,其他各参数设置如表 1 所示,其中,空白代表不进行更改设定。

表 1　4 种负载的参数

分类	fileserver	makedirs	listdirs	createfiles
文件个数	10 000		10 000	30 000
目录个数		1 000 000		
平均文件宽度	20	100	5	100
平均文件大小/KB	2			2
I/O 大小/MB	1			1
平均更新大小/KB	16			

3.3　测试结果及分析

使用 filebench 进行性能测试,可以获得操作处理速度和 I/O 带宽 2 个测试结果,它们的测试结果分别如图 8 和图 9 所示。

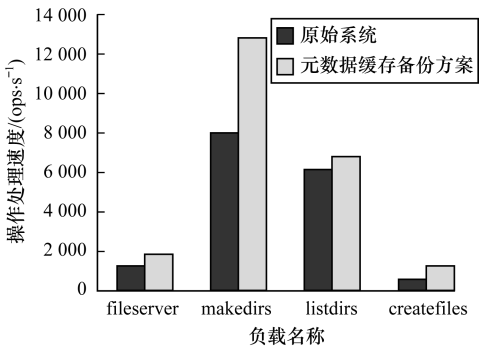

图 8　不同负载的操作处理速度

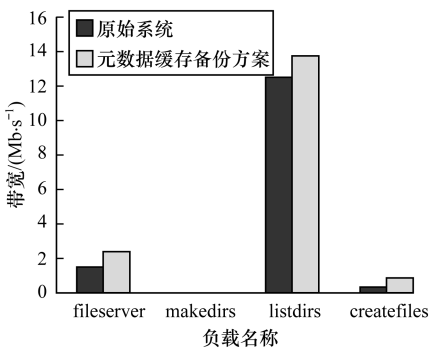

图 9　不同负载的 I/O 带宽

根据图 8 和图 9 可知,使用 fileserver,makedirs,listdirs 和 createfiles 4 种负载,相比于 Ceph 系统的原

元数据管理子系统,元数据缓存备份方案能有效提升操作处理速度和 I/O 带宽。在 fileserver 负载中都是小文件,元数据访问请求比例占该负载的所有访问请求的比例非常大,makedirs,listdirs 和 createfiles 3 种负载,主要都是元数据访问请求,所以,相比于原元数据管理子系统,元数据缓存备份方案在能够提升元数据访问性能的前提条件下,就能够提升表 1 所述的 4 种负载的性能。

需要注意的是,makedirs 负载只有元数据操作,没有数据读写操作,所以,在图 9 中,makedirs 负载的带宽为 0。

fileserver,makedirs,listdirs 和 createfiles 4 种负载都是采用的 8 个线程测试的,为了测试不同线程数目中,元数据缓存备份方案的性能情况,基于 listdirs 负载,分别测试 1 个线程、2 个线程、4 个线程、8 个线程和 16 个线程的 listdirs 的性能,测试结果如图 10 和图 11 所示。

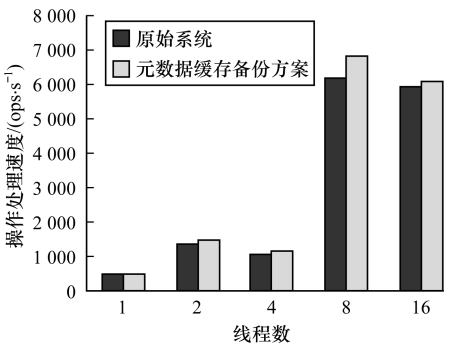

图 10 不同线程数下的操作处理性能

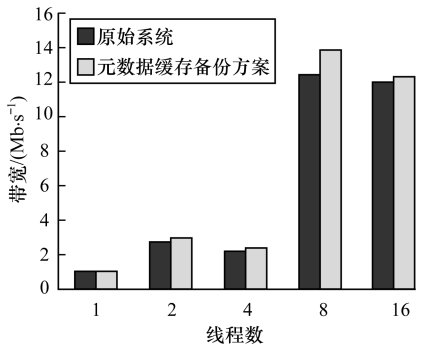

图 11 不同线程数下的 I/O 带宽

相比于原元数据管理子系统,元数据缓存备份方案当线程数目为 1 和 16 时,操作处理速度和 I/O 带宽几乎没有提升,这是因为,当线程数目为 1 时,整个 Ceph 文件系统的带宽还没有提升起来,当线程数目为 16 时,整个 Ceph 系统达到了最大处理并发度;当线程数目为 2,4 和 8 时,操作处理速度和 I/O 带宽性能都有所提升。相比于原元数据管理子系统,元数据缓存备份方案在线程数目为 2 时,操作处理速度和 I/O 带宽分别提升了 6.7% 和 7.1%,在线程数目为 4 时,操作处理速度和 I/O 带宽分别提升了 8.1% 和 9.1%,在线程数目为 8 时,操作处理速度和 I/O 带宽分别提升了 10.6% 和 11.2%。

4 结束语

本文基于 Ceph 系统,以元数据管理子系统为研究对象,就如何提升元数据的访问性能展开研究工作,主要研究了基于 Ceph 文件系统的元数据缓存备份技术。在 Ceph 系统中,元数据服务器集群负责维护元数据的管理,为提升整个系统的元数据访问性能,基于 Ceph 系统提出了元数据缓存备份方案,为减少元数据访问操作中元数据信息更新到 OSD 集群所需时间,将 MDS 节点中的缓存的元数据信息备份到其他 MDS 节点中,从而能够在保证系统安全性前提下,有效降低元数据响应延时。使用 filebench 测试工具,测试了元数据缓存备份方案和 Ceph 系统原元数据管理子系统的性能。测试结果表明,相比于原元数据管理子系统,元数据缓存备份方案能够有效提升元数据的访问性能。

参考文献

[1] Braam P J. The Coda Distributed File System[J]. Linux Journal,1998,50(6):10-20.

[2] Fetterly D, Haridasan M, Isard M, et al. TidyFS: A Simple and Small Distributed File System [C]// Proceedings of USENIX Annual Technical Conference. Berkeley,USA:USENIX,2011:34-48.

[3] Kubiatowicz J,Bindel D,Chen Y,et al. Oceanstore:An Architecture for Global-scale Persistent Storage [J]. ACM Sigplan Notices,2000,35(11):190-201.

[4] Weil S A,Brandt S A,Miller E L,et al. Ceph:A Scalable, High-performance Distributed File System [C]// Proceedings of the 7th Symposium on Operating Systems Design and Implementation. Berkeley, USA: USENIX, 2006:307-320.

[5] Schmuck F B,Haskin R L. GPFS: A Shared-disk File System for Large Computing Clusters[C]//Proceedings of Conference on File and Storage Technologies. Berkeley,USA:USENIX Association,2002:19-33.

[6] Ghemawat S, Gobioff H, Leung S T. The Google File System[C]//Proceedings of ACM SIGOPS Operating Systems Review. New York,USA:ACM Press,2003:29-43.

[7] Shvachko K, Kuang H, Radia S, et al. The Hadoop Distributed File System[C]//Proceedings of the 26th IEEE Symposium on Mass Storage Systems and Technologies. Washington D. C. ,USA:IEEE Press,2010:1-10.

[8] Miller E L,Brandt S A,Long D D E. HeRMES:High-performance Reliable MRAM-enabled Storage [C]// Proceedings of the 8th Workshop on Hot Topics in Operating Systems. Washington D. C., USA: IEEE Press,2001:95-99.

(下转第 83 页)

离超过 100 m 时,丢包开始增加,110 m 时的丢包率为 4% 。当通信距离最远达到 260 m,仍然可以探测到信号,但是通信效果非常不好,线路时断时连,同时丢包率严重升高。测试中改变串口波特率,分别设置为 115 200 b/s 和 38 400 b/s,但波特率的改变并没有影响丢包率的变化。加入功率放大器 CC2591 后,延长了通信距离且提高了稳定效果,相同距离丢包率明显下降,在 1 000 m 以内可以保证数据的准确传输,此距离能够满足微电网控制信息传输的需求。由于实验过程中存在行人阻碍、天气变化、附近干扰源等的影响,某些测量数据存在误差。

5　结束语

ZigBee 无线通信具有网络容量大、传输速率相对较高、占用频段免费的优点,可以作为分布式发电微电网信息传输的载体。本文提出 ZigBee 无线通信微电网监控网络,采用将虚拟地址与传输数据绑定的方式,使微电网监控网络避免因动态分配地址对现场设备地址识别不准确的问题。采用节点就近选择能量峰值高的信道入网,可以提高通信链路的可靠性。实验结果表明,组建的 ZigBee 网络能够稳定传输数据,地址识别准确可靠。本文设计的 ZigBee 无线通信监控网络能够满足分布式发电微电网监控信息传输的要求,增加微电网系统的开放性和易扩展性。

参考文献

[1] 陈　新,姬秋华,刘　飞. 基于微电网主从结构的平滑切换控制策略[J]. 电工技术学报,2014,29(2):163-170.

[2] 高春凤,杨仁刚,王江波. 基于虚拟频率的微电网下垂控制策略设计[J]. 电网技术,2013,37(12):3331-3335.

[3] 张庆海,彭楚武,陈燕东. 一种微电网多逆变器并联运行控制策略[J]. 中国电机工程学报,2012,32(25):126-132.

[4] Khorsandi A, Ashourloo M, Mokhtari H. Automatic Droop Control for a Low Voltage DC Microgrid[J]. IET Generation, Transmission and Distribution, 2016, 10(1): 41-47.

[5] 王　毅,张丽荣,李和明. 风电直流微电网的电压分层协调控制[J]. 中国电机工程学报,2013,33(4):16-24.

[6] Thale S, Wandhare R, Agarwal V. A Novel Reconfigurable Microgrid Architecture with Renewable Energy Sources and Storage[J]. IEEE Transactions on Industry Applications, 2015, 51(2): 1805-1816.

[7] 李　哲,刘　澄,徐石明. 微电网协调控制过程中 EtherCAT 总线的应用[J]. 电力系统自动化,2012,36(24):39-43.

[8] 徐焜耀,徐　鑫,侯兴哲. 构建新一代智能配用电通信网建议[J]. 电力系统自动化,2013,37(10):1-5.

[9] 王　雪,钱志鸿,李　冰. 蓝牙自适应分组选择策略与选择重传算法研究[J]. 通信学报,2011,32(1):151-158.

[10] 廖鹏飞,陈庆奎. 基于蓝牙 4.0 到 3G 的无线传感器网关设计与实现[J]. 计算机工程,2015,41(9):13-18.

[11] 牛建伟,刘　洋,卢邦辉. 一种基于 Wi-Fi 信号指纹的楼宇内定位算法[J]. 计算机研究与发展,2013,50(3):568-577.

[12] 徐潇潇,谢林柏,彭　力. 基于 WiFi 信号强度特征的室内定位系统设计[J]. 计算机工程,2015,41(4):87-91.

[13] Ahmed M, Kang Y, Kim Y. Communication Network Architectures for Smart-house with Renewable Energy Resources[J]. Energies, 2015, 8(8): 8716-8735.

[14] 唐伟华,石高涛. ZigBee 网络中基于概率 CSMA 的 WiFi 干扰避免方法[J]. 计算机工程,2016,42(4):55-59.

[15] Chen Yu-kai, Wu Yung-chun, Song Chau-chung, et al. Design and Implementation of Energy Management System with Fuzzy Control for DC Microgrid Systems[J]. IEEE Transactions on Power Electronics, 2013, 28(4): 1563-1570.

编辑　刘　冰

(上接第 72 页)

[9] Edel N K, Tuteja D, Miller E L, et al. MRAMFS: A Compressing File System for Non-volatile RAM[C]// Proceedings of the IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems. Washington D. C., USA: IEEE Press, 2004: 596-603.

[10] Doh I H, Choi J, Lee D, et al. Exploiting Non-volatile RAM to Enhance Flash File System Performance[C]// Proceedings of the 7th ACM & IEEE International Conference on Embedded Software. New York: ACM Press, 2007: 164-173.

[11] Doh I H, Lee H J, Moon Y J, et al. Impact of NVRAM Write Cache for File System Metadata on I/O Performance in Embedded Systems[C]//Proceedings of the 2009 ACM Symposium on Applied Computing. New York, USA: ACM Press, 2009. 1658-1663.

[12] Suk J, No J. HybridFS: Integrating NAND Flash-based SSD and HDD for Hybrid File System[C]//Proceedings of the 10th WSEAS International Conference on Applied Informatics and Communications. New York, USA: ACM Press, 2010: 178-185.

[13] Wang A I A, Kuenning G, Reiher P, et al. The Conquest File System: Better Performance Through a Disk/persistent-ram Hybrid Design[J]. ACM Transactions on Storage, 2006, 2(3): 309-348.

[14] 刘扬宽. 分布式文件系统中新型元数据管理子系统的研究与实现[D]. 镇江: 江苏大学,2012.

[15] 李　帅. 分布式文件系统中基于非易失存储器的缓存子系统[D]. 镇江: 江苏大学,2013.

编辑　刘　冰