全闪存 Ceph 的混合优化策略研究

李 博[1] 颜靖艺[2]

[1](桂林航天工业学院计算机科学与工程学院 广西 桂林 541004)

[2](桂林信息科技学院商学院 广西 桂林 541004)

摘 要 Ceph 是一种经典的分布式存储系统,但随着数据量级和实验要求的不断提高,Ceph 在读写延迟方面的不足制约了其进一步的发展。随着固态硬盘(Solid State Disk,SSD)等新型存储介质与 Ceph 的结合,Ceph 整体读写性能有了明显的提升。全闪存 SSD 是一种性能优秀的 SSD,基于全闪存 SSD 的 Ceph 系统表现也更好。与传统的分布式系统的读写延迟制约于后端存储介质不同,在基于全闪存 SSD 的情况下,Ceph 系统的软件栈也需要优化以匹配硬件读写速度的提升。因此,提出基于全闪存 SSD 的 Ceph 存储系统的混合优化策略,针对 Ceph 的加锁机制、Crush 算法等方面进行了优化,实验结果表明性能优化效果大约在 2.8% ~4.5% 之间。

关键词 Ceph NVMe SSD Crush 算法 BlueStore

中图分类号 TP311 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2024.06.019

MIXED OPTIMIZATION STRATEGY OF NVME SSD CEPH

Li Bo[1] Yan Jingyi[2]

[1](*School of Computer Science and Engineering*, *Guilin University of Aerospace Technology*, *Guilin 541004*, *Guangxi*, *China*)

[2](*Business School*, *Guilin Institute of Information Technology*, *Guilin 541004*, *Guangxi*, *China*)

Abstract Ceph is a classic distributed storage system, but with the continuous improvement of data and experimental requirements, the lack of read-write delay of Ceph restricts its further development. With the combination of Ceph and new storage media, such as solid state disk (SSD), the overall read-write performance of Ceph has been significantly improved. Non-volatile memory express (NVMe) SSD is an excellent SSD, and Ceph system based on NVMe SSD performs better. Different from the traditional distributed system, the read-write delay of NVMe SSD Ceph isn't limited by hardware. This paper considers that in the case of NVMe SSD the software stack of Ceph also needs to be optimized to match the improvement of hardware read-write speed. Therefore, this paper proposes a hybrid optimization strategy for NVMe SSD Ceph: in terms of the locking mechanism of Ceph, Crush algorithm etc. The experimental results show that the improvement effect is about 2.8% ~4.5%.

Keywords Ceph NVMe SSD Crush algorithm BlueStore

0 引 言

据 International Data Corporation 预计,到 2025 年时世界数据总量将可能达到 175 ZB。随着数据量级的不断增长,如何高效稳定存储如此大规模的数据,成为目前一个热点研究问题。在诸多解决方案中,选用分布式存储系统是处理大规模数据存储的一种常用手段。分布式存储系统基于多台存储服务器分摊数据量,采用备份和寻址算法保证了鲁棒性和精确性,从而可以高效可靠地存储大规模数据。目前常见的分布式存储系统有 HDFS(Hadoop Distributed File System)和 Ceph 等,而随着数据量级和实验要求的不断提高,研究人员也在继续探索硬件配置或软件代码等方面进一步提升分布式存储系统性能的可能性。

Ceph 是一种经典的分布式存储系统,相比于其他

收稿日期:2020 - 12 - 08。桂林航天工业学院校级科研基金项目(XJ20KT20)。**李博**,助教,主研领域:数据挖掘和云存储。**颜靖艺**,讲师。

系统,Ceph 具有可靠性高、灵活性强、可扩展性高等优点,并且代码开源,方便研究人员开发。但是传统的 Ceph 系统受制于读写延迟,难以适应数据量级的进一步增长。固态硬盘(Solid State Disk,SSD)是以闪存作为永久存储的一种新型存储介质,相比于传统的 HDD(Hard Disk Drive),SSD 读写性能有了明显的提升,其中,NVMe SSD 是一种新型高效的 SSD。Ceph 原本是为传统的 HDD 而设计,HDD 硬件读写速度较慢,因此软件栈参数设置导致的一些延迟对于整体系统的读写延迟并没有太多影响。但随着硬件存储介质的优化,对于采用新型存储介质的 Ceph 文件系统,尤其是在基于全闪存 SSD 的情况下,硬件的读写不再是制约系统读写延迟的主要因素,软件栈的性能也对系统读写性能有着重要影响。因此,本文尝试以全闪存 SSD 的存储介质来探索 Ceph 文件系统的瓶颈所在,研究如何对 Ceph 的软件栈进行优化,以匹配新型存储介质[1]。

1　Ceph 系统概述

Ceph 是目前常用的分布式文件系统之一,可支持大规模的云数据存储,其基于 Crush 算法,性能随着节点数目的增加不会有明显降低。Ceph 在存储数据时,也会尽量保持各个节点的数据负载均衡,并计算出其数据存储的位置。在 Ceph 中,可靠、自动、分布式对象存储(RADOS)是 Ceph 存储集群的基础。Ceph 中的一切都以对象的形式存储,而 RADOS 就负责存储这些对象,而不考虑数据类型。本节将对 Ceph 存储系统架构、BlueStore 和 Crush 算法进行相应的介绍[5]。

1.1　Ceph 分布式存储系统架构

Ceph 的基础架构如图 1 所示,Ceph 是由 RADOS、LIBRADOS、RADOSGW、RBD 和 CEPH FS 组成,其中最核心的就是 RADOS。如果把 RADOS 剖离出来,它也是一个可靠、智能的分布式对象存储系统。RADOS 系统是由 OSD 和 Monitor 组成,负责 Ceph 中数据的存储,并有很好的性能。从图 1 中可以看到,与 RADOS 层直接相连的是 LIBRADOS,LIBRADOS 本质是一个库(Library),支持多种语言编写的应用程序访问 RADOS 系统。而在 LIBRADOS 层之上还有三层:分别是 RADOSGW、RBD 和 CEPH FS。RADOSGW 层相当于网管的作用,并有一定的兼容性。RBD 是通过 Linux 内核客户端和 QEMU/KVM 驱动来提供一个分布式的块设备。CEPH FS 通过 Linux 内核客户端和 FUSE 来提供一个兼容 POSIX 的文件系统。

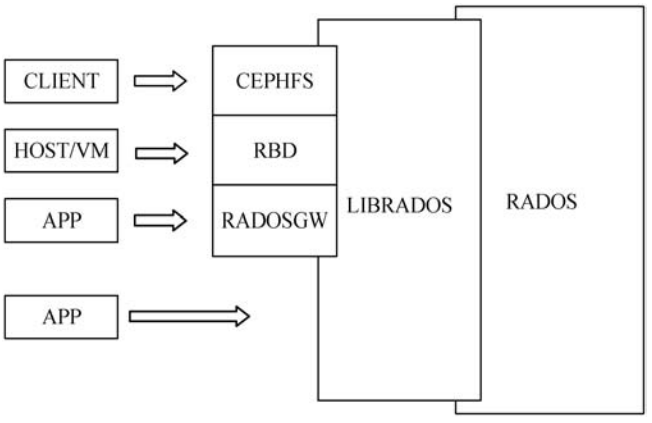

图 1　Ceph 存储系统架构

1.2　BlueStore

传统的 Ceph 的对象存储引擎采用的是 FileStore,但是 FileStore 存在写放大的问题,并且在操作中还要经过操作系统的通用文件系统层,因此 FileStore 的性能也是制约 Ceph 的瓶颈之一。针对 FileStore 的缺陷,Ceph 在 Jewel 版本之后推出了构建在裸磁盘设备之上的 BlueStore,并且也提高了对于新存储设备的适应性,图 2 是 BlueStore 的整体架构。BlueStore 的出现提升了 Ceph 的性能,并且充分提升了对 SSD、NVMe 等新型存储介质的匹配性。由于 BlueStore 对数据的直接可操作性、新型存储介质的适应性,BlueStore 成为目前 Ceph 的主流的存储引擎,本文也将对基于 BlueStore 存储引擎的 Ceph 系统进行优化。

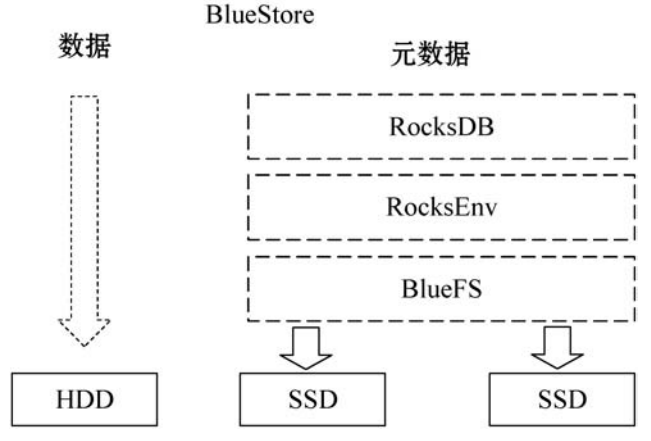

图 2　BlueStore 整体结构

1.3　Crush 算法

在 Ceph 存储中,数据都是以 Object 为基本单位进行存储的,每个 object 默认为 4 MB 大小,若干个 Object 属于一个归置组(Placement Group,PG),而若干个 PG 又属于一个 OSD(Object Storage Device),一般来说,一个 OSD 对应于一块磁盘。Ceph 采用层级化的集群结构(Hierarchical Cluster Map),并且用户可自定义该集群结构,而 OSD 正是这个层级化集群结构的叶子节点。

Ceph 是采用 CRUSH(Controlled Replication Under Scalable Hashing)算法来进行数据的分配和查找,Crush 算法是基于哈希的数据分布算法,其每个 objec 模块通常定义为 4 MB,本文通过实验分析这里存在优化的空间,后续也就此进行相应的优化。

在进行 SSD 等新型存储介质的研究时,由于 SSD 较普通磁盘更为昂贵,以及为了方便调试内部参数,科学工作者通常会选用 SSD 的模拟器来进行相应的研究。目前,常用的 SSD 模拟器主要有 FlashSim、Disksim + ssdextension 和 NANDFlashSim。这三种模拟器各有其相应的优势,可以进行算法的模拟和 SSD 中相关参数的设计,在实验中探究读写响应时间,从而节省实验成本。Ceph 是一个成熟的分布式文件系统,其中还有一些其他重要的组成模块,本文碍于篇幅,以及其他模块并没有选为优化的重点,故不继续进行介绍[6-7]。

2 Ceph 系统的优化策略

2.1 现有的一些优化策略

目前,有很多科学工作者对 Ceph 的优化进行多个角度的探索。比如:搭建 SSD 和传统磁盘的混合存储系统,这种方式已经应用到实际中;研究人员分析全闪存条件下对 Ceph 的运行原理进行优化;优化 Ceph 存储引擎和使用动态负载均衡策略等。这些方法从各个角度对 Ceph 提出了优化的思路,本文也会借鉴这些思路,继续进行相应的研究。

Sage Weil 在最开始设计和应用 Ceph 分布式存储系统时,主要针对的是普通的 HDD 盘。也就是通过合理配置多台廉价的计算机可以超过单台性能优越的计算机。并且,传统思维普遍认为硬件延迟会比软件延迟高出数个量级,软件的设计允许并不完美。因此 Ceph 文件系统最初软件栈的设计并没有考虑引入 SSD 并产生的变化。随着 SSD 的降价和推广,以及更为新型存储器件的出现,相比于传统的 HDD,硬件端的读写性能在不断提升。在这种情况下,科学人员需要转换思维,软件读写延迟相比于硬件读写延迟不能再完全忽略不计,因此 Ceph 软件栈的性能也需要相应的提升,以匹配硬件性能的优化[2-4]。

2.2 混合存储——全闪存 SSD

随着 SSD 等新型存储介质的价格不断趋向于平民化,可以预见,之前受价格影响所导致的 SSD 等新型存储介质的应用的局限性将被打破,更多的分布式存储系统将采用混合存储的方式,甚至于全 SSD 的存储。本文依据这一思路进行更为大胆的尝试,采用全 NVMe SSD 存储,分析 Ceph 软件栈的瓶颈所在。非易失性存储器标准(Non-Volatile Memory Express,NVMe)SSD 是使用 PCI-E 通道的 SSD,相比于普通的 SSD,NVMe SSD 延迟更低,IOPS 大增,功耗更低,适用范围更广。虽然 NVMe SSD 价格相比于 SSD 又高出了很多,但本文认为随着制造工艺的提升,价格将不再是制约大数据存储的主要因素。

NVMe 的低延迟性,是因为流线型的存储堆栈,这样 NVMe 可以更快地发布命令,充分利用了 SSD 设备的并行性,提升了 SSD 的随机性能。而在采用 SSD 或者 NVMe SSD 替代传统的 HDD 后,硬件的读写性能将会明显提升,这时就需要考虑网络和协议栈所产生的延迟[8-9]。

2.3 日志资源池化

日志功能是所有存储系统中都很重要的一个功能,Ceph 也给用户提供输出日志的功能,并且在 Ceph 的配置文件中进行相关参数的配置。通过阅读 Ceph 源码,Ceph 中有一个 log_Entry 对象,在 Ceph 在进行日志相关的操作时,将会反复涉及 log_Entry 对象的申请和调用。而 log_Entry 对象频繁地申请释放,这毫无疑问会对系统的性能产生一定的影响。本文希望通过对 log_Entry 对象进行管理,从而使得该对象在日志操作过程中的使用更为合理。

本文进行相应管理的操作是在 Ceph 系统初始化时,创建多个 log_Entry 对象,并保存在一个数据池当中。在进行日志操作需要调用 log_Entry 对象时,就不需要关闭之前的申请,允许多个 log_Entry 对象的存在,使得申请释放的开销降低。然后定时关闭多余的 log_Entry 对象,避免资源的浪费。

2.4 加锁机制优化

在数据库系统中,锁机制是维护系统稳定性的一个重要机制,系统要实现事务管理,锁机制发挥着重要的作用。在基于 HDD 等传统存储介质的情况下,加锁和解锁所耗费的时间相比于硬件读写的时间是微不足道的。但在全闪存的情况下,锁机制所花费的时间也将不能再忽视。Ceph 作为一个成熟稳定的系统,设计了非常详细的锁机制。由于 Ceph 在使用 Crush 算法处理数据 I/O 时不需要元数据服务器和客户端之间的通信,因此只需研究 OSD 和客户端在处理写入和读取时发生的通信,而不是整体 I/O 路径,这样简化了研究的流程。

PG 锁在 Ceph 中非常常见,很多操作都需要 PG

锁定,比如日志写入等;并且 PG 锁对于每个 I/O 请求也会保持锁定较长的时间。因此本文设计将这部分的锁机制进行优化,Ceph 中默认的 OP 线程是 2 个,本文设计将其增加为 32 个,这样就会大大降低了 PG 锁对于整个系统的延迟影响。

2.5　调整 Ceph 中 chunk 大小

如前所述,Crush 算法是 Ceph 中用于查找匹配数据的重要算法,因此 Crush 算法性能的好坏对 Ceph 整体性能有着显著的影响。本文研究发现,通过对 Ceph 进行测试,发现 Ceph 的连续读写性能是不及 Ceph 的随机性能的。本文推测可能是在顺序读写的过程中,Crush 算法中设置 chunk 的大小不当而对性能造成影响,比如顺序写时由于 chunk 太小,导致 chunk 的数量过多从而降低消息的传输效率,导致性能下降。Ceph 的 Crush 算法将 chunk 的大小设置为 4 MB,本文通过将其改为 1 MB 和 2 MB,进行相应的分析,比较 chunk 大小是否会对 Ceph 的性能造成影响。

在前文基础上,本文也会尝试这几种优化方法组合起来,以混合优化策略再进行实验分析,从而探究在全 NVMe SSD 下对 Ceph 最为适合的优化方案。

3　实　验

目前,新型存储介质有着优秀的读写性能(SSD 相比于 HDD 读写延迟有明显减少),而随着网络带宽的不断增加,制约大数据存储发展的因素也发生了变化。传统思维中的简单更换高速存储设备有些并不能产生明显的性能提升,在这种情况下,Ceph 软件栈也需要相应的优化。本文通过将 SSD 进一步替换为更高速的全闪存 SSD,来探讨 Ceph 软件栈的速度瓶颈,从而进一步提升 Ceph 的性能[5]。本节继续前面章节的思路,搭建实验环境进行分析。

3.1　测试工具与测试环境介绍

本文主要针对 Ceph RBD 设备进行测试,测试了在块大小为 4 K、8 K、16 K、32 K、64 K 和 128 K 时,顺序读、随机读、顺序写和随机写的 IOPS 和延迟,并对比分析它们之间的性能表现。实验中共用了 5 台服务器,其中 4 台作为 Ceph 的服务器,1 台作为客户端。其中服务器的型号为 Sugon S650,CPU 为 AMD Opteron(TM) 6212,CPU 频率为 2.6 GHz,核心数为 16 核,内存大小为 64 GB,操作系统为 Ubuntu 16.04,Linux 内核版本为 4.4.0-124-generic。实验所用的 Ceph 的版本为社区 12.2.4 版本。

3.2　测试指标

本文对于主要从 IOPS、延迟两个方面对 Ceph RBD 块设备进行测试分析。IOPS(I/Os Per Second)代表的是每秒钟发生的 I/O 的次数;延迟是系统延迟的度量,对于存储系统而言,延迟则是指响应 I/O 请求所需的时间,通常是每一次 I/O 操作的平均花费的时间。这两个指标从不同的方面反映了 Ceph 块设备的性能,本节实验主要是针对 IOPS 和延迟进行测试分析。

3.3　日志资源池化

根据本文第 2.3 节的分析,日志功能会对 Ceph 系统的性能有一定的影响。本文进行的改进是在 Ceph 系统初始化时,创建多个 log_Entry 对象,并保存在一个数据池当中。在进行日志操作需要调用 log_Entry 对象时,就不需要关闭之前的申请,允许多个 log_Entry 对象的存在,使得申请释放的开销降低。然后定时关闭多余的 log_Entry 对象,避免资源的浪费。

通过这种日志资源优化方式,对于不同的负载模型,都是有一定的优化效果,图 3 - 图 4 显示:性能提升幅度在 1% ~3% 之间。所以,日志资源池化这个优化策略值得继续研究。

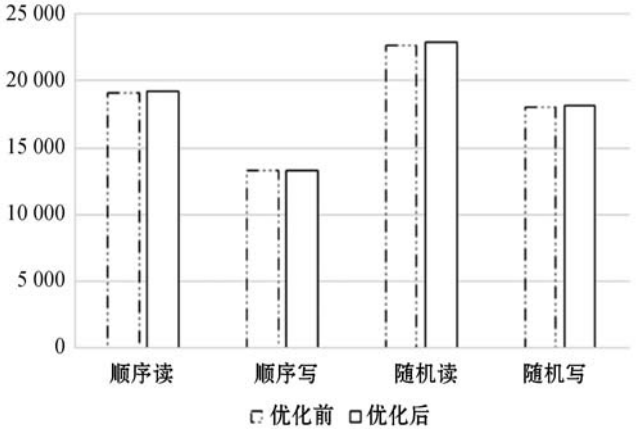

图 3　日志资源池化方式对 IOPS 的优化对比

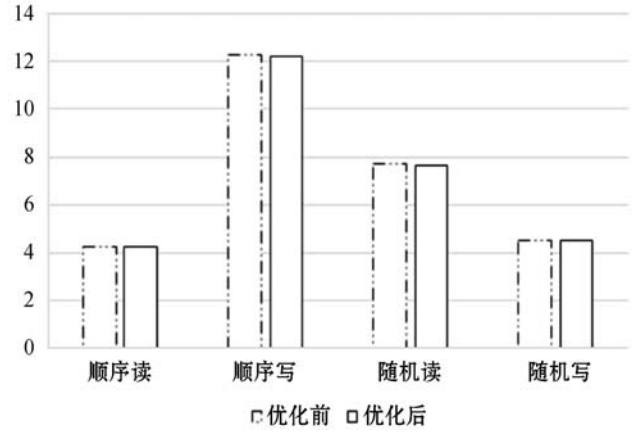

图 4　日志资源池化方式对延迟(LAT)的优化对比

3.4 加锁机制优化

本文分析 PG 锁在 Ceph 中日志写入操作时会发挥重要的作用,以及在每个 I/O 请求中,PG 锁也会保持锁定较长的时间。因此本文设计将这部分的锁机制进行优化,Ceph 中默认的 op 线程是 2 个,本文设计将其增加为 32 个,这样就会大大降低了 PG 锁对于整个系统的延迟影响。图 5 – 图 6 显示:除顺序写和随机写外,IOPS 提升 1% ~3%,延迟缩短 4% ~7%,实验说明,本文提出的加锁机制优化策略对于 PG 锁的优化有一定的优化效果,尤其是延迟(LAT)方面。

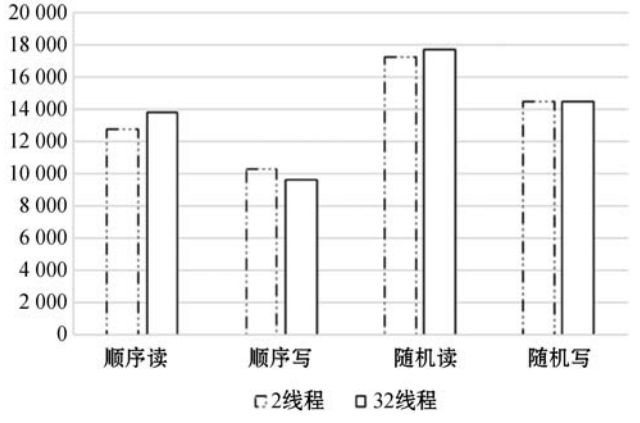

图 5 加锁机制对 IOPS 的优化对比

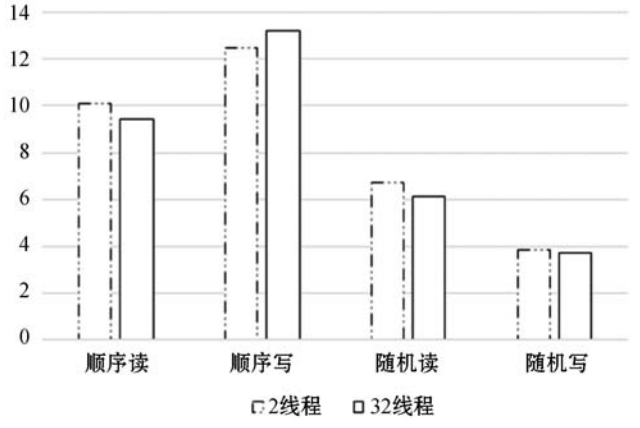

图 6 加锁机制对延迟(LAT)的优化对比

3.5 调整 Ceph 中 chunk 大小

Ceph 是采用 Crush 算法来查找匹配数据,所以提升 Crush 算法的性能,也会提升 Ceph 的整体性能。本文通过实验对比发现研究发现 Ceph 的连续读写性能与 Ceph 的随机读写性能存在一定差异。根据 2.5 节的分析,本文通过改变 Crush 算法中 chunk 的大小来进行分析,比如顺序写时由于 chunk 太小,导致 chunk 的数量过多从而降低消息的传输效率,导致性能下降。Ceph 的 Crush 算法将 chunk 的大小设置为 4 MB,本文在这里进行相应的实验探索,通过将 chunk 大小设置为 1 MB 和 2 MB,研究 chunk 大小是否会对 Ceph 的性能造成影响。

如图 7 和图 8 所示,本文在实验中将 chunk 大小分别设置为 4 MB、2 MB 和 1 MB,并将实验结果进行对比。可以看出 chunk 大小的修改对顺序写有 3% 左右的提高,说明这一思考方向值得进一步分析。

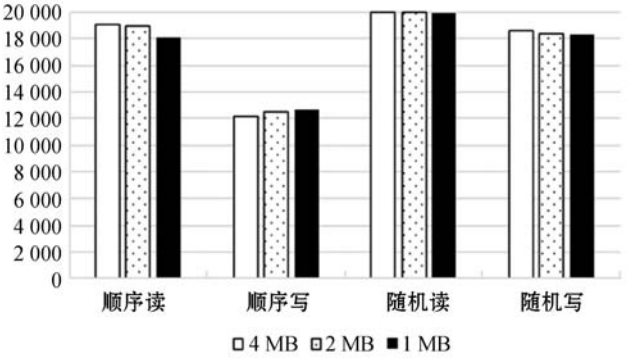

图 7 chunk 大小对 IOPS 的影响对比图

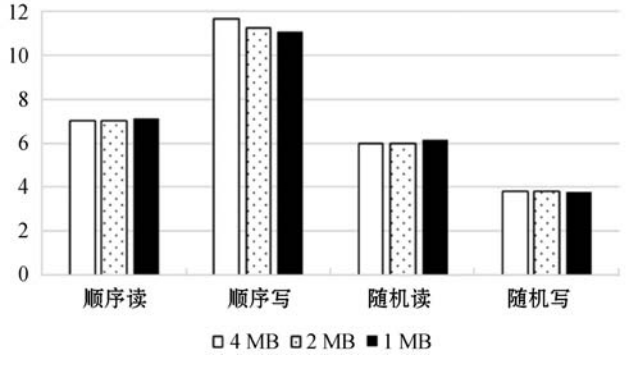

图 8 chunk 大小对延迟(LAT)的影响对比

3.6 混合优化

本文在 2.1 节对采用 NVMe SSD 的 Ceph 进行了分析,然后在 2.2 节 – 2.4 节依次提出了一些改进的思路,并依次在 3.3 节 – 3.5 节进行实验分析。通过前几节的实验,说明这些方法对于 Ceph 的性能提升都有一定的效果。本文现在尝试将这些优化的方案组合起来,形成混合优化策略,来探索是否有更好的优化效果。具体实验操作是将不同的优化方案组合到一起,比如日志资源池化、加锁机制优化和调整 chunk 大小等,然后在此基础上对 Ceph 集群的性能进行测试,然后和优化前的基准性能进行对比,找出在全 NVMe SSD 下对 Ceph 较为适合的优化方案。

本文对于日志池化、加锁机制优化以及调整 chunk 大小进行了组合测试,图 9 – 图 10 为不同优化方案组合之后 Ceph 集群性能测试结果,组合其中多项优化策略进行优化,从测试结果中的优化结论可以看出组合多项优化策略 Ceph 的优化效果。几组调小 chunk 的混合优化策略组合对顺序写、顺序读和随机写的延迟(LAT)有较好的优化效果,并且这三种优化方式一起组合时,优化效果更为明显,性能优化效果大约在 2.8% ~4.5% 之间。

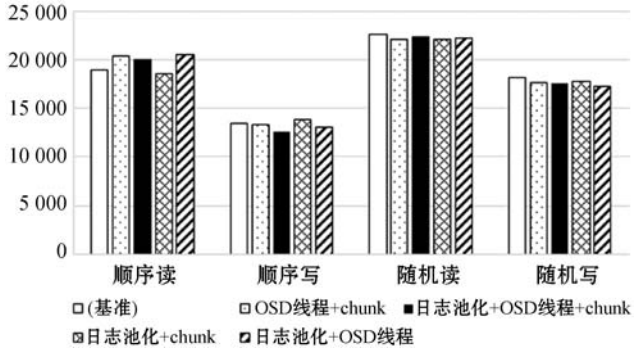

图 9　混合优化方式对 IOPS 的优化对比

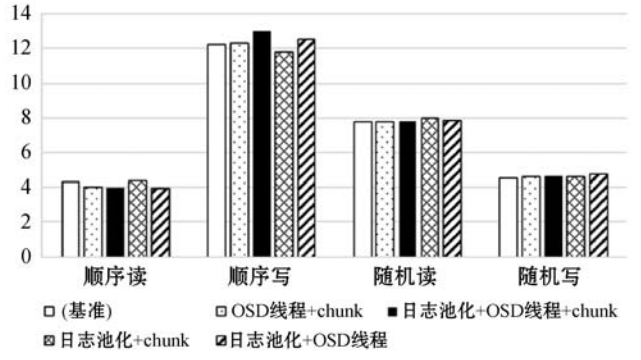

图 10　混合优化方式对延迟(LAT)的优化对比

4　结　语

大数据存储是目前的热点研究方向之一,对于科学技术的发展有着重要意义,分布式存储是解决大数据存储问题的一种有效手段。Ceph 作为一种非常经典成熟的分布式系统,在传统基于 HDD 的硬件存储中有非常优秀的表现。但随着 SSD 等新型存储介质的不断涌现,如何进一步提升分布式存储系统的性能,Ceph 原有的软件栈是否还可以匹配新型存储介质的高速读写速度,非常值得科学工作者的研究。本文以前瞻性的角度,大胆采用全 NVMe SSD 作为硬件存储介质,分析 Ceph 软件栈所造成的读写延迟相比于新型存储介质的读写延迟不能再忽略不计,故提出了混合优化策略。混合优化策略包括 log 资源池化以减少 log Entry 频繁创建与释放的开销,加锁机制的优化来减少 PG 锁的阻塞,每块 NVMe SSD 设置多个 OSD 分区以减少锁争用,调整系统中 chunk 的大小以达到最优的效果。并且还组合了以上几种优化方案进行对比测试,本文认为这三种优化方式进行组合的优化效果最明显。

虽然本文对于 Ceph 优化做了很多相关研究,但还有一些可以提高的地方,比如优化性能进一步提升、实验适用场景进一步扩展等。今后会继续深入相关研究,以进一步优化 Ceph 等分布式存储系统的性能,推进大数据存储研究的发展。

参　考　文　献

[1] 沈志荣,薛巍,舒继武. 新型非易失存储研究[J]. 计算机研究与发展,2014,51(2):445 - 453.

[2] Yamato Y. Cloud storage application area of HDD-SSD hybrid storage, distributed storage, and HDD storage [J]. IEEJ Transactions on Electrical and Electronic Engineering, 2016,11(5):674 - 675.

[3] Oh M, Eom J, Yoon J, et al. Performance optimization for all flash scale-out storage[C]//IEEE International Conference on Cluster Computing,2016:316 - 325.

[4] Lee D Y, Jeong K, Han S H, et al. Understanding write behaviors of storage backends in Ceph object store[C]//IEEE International Conference on Massive Storage Systems and Technology,2017.

[5] 董聪,张晓,程文迪,等. 基于新型存储器件的分布式文件系统性能优化[J]. 计算机应用,2020,40(12):3594 - 3603.

[6] Song X, Yang J, Chen H B. Architecting flash-based solid-state drive for high-performance I/O virtualization[J]. IEEE Computer Architecture Letters,2014,13(2):61 - 64.

[7] Kim B, Park M, Jeon C, et al. AAGC: An efficient associativity-aware garbage collection scheme for hybrid FTLs[C]//27th Annual ACM Symposium on Applied Computing,2012:1785 - 1790.

[8] Laliberte B. Automate and optimize a tiered storage environment-FAST! [EB/OL]. [2020 - 12 - 08]. https://wenku.baidu.com/view/1a4d651902768e9951e738af.html.?_wkts_=1713151182549.

[9] Lin L, Zhu Y F, Yue J H, et al. Hot random off-loading: A hybrid storage system with dynamic data migration[C]//19th International Symposium on Modeling,2011:318 - 325.

(上接第 107 页)

[40] Huang G, Zhu Q, Siew C. Extreme learning machine: A new learning scheme of feedforward neural networks[C]//2004 IEEE International Joint Conference on Neural Networks,2004:985 - 990.

[41] Huang G, Zhu Q, Siew C. Extreme learning machine: Theory and applications[J]. Neurocomputing,2006,70(1):489 - 501.

[42] Huang G, Chen L, Siew C. Universal approximation using incremental constructive feedforward networks with random hidden nodes[J]. IEEE Transactions on Neural Networks, 2006,17(4):879 - 892.

[43] Huang G, Zhou H, Ding X, et al. Extreme learning machine for regression and multiclass classification[J]. IEEE Transactions on Systems, Man and Cybernetics, Part B, 2012,42(2):513 - 529.