计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(2):108−113 [doi: 10.15888/j.cnki.csa.008322]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Ceph 存储系统的小文件存储优化方案①

陈法河, 柴小丽

(中国电子科技集团 第三十二研究所, 上海 201808)
通信作者: 陈法河, E-mail: fahechen@163.com

摘　要: 针对 Ceph 存储系统面对小文件存储时存在元数据服务器性能瓶颈、文件读取效率低等问题. 本文从小文件之间固有的数据关联性出发, 通过轻量级模式匹配算法, 提取出关联特征并以此为依据对小文件进行合并, 提高了合并文件之间的合理性, 并在文件读取时将同一合并文件内的小文件存入客户端缓存来提高缓存读取命中率, 经过实验验证本文的方案有效的提高了小文件的访问效率.

关键词: 分布式存储; 小文件存储; 数据关联性; 文件合并

引用格式: 陈法河,柴小丽.基于 Ceph 存储系统的小文件存储优化方案.计算机系统应用,2022,31(2):108–113. http://www.c-s-a.org.cn/1003-3254/8322.html

Optimization of Small File Storage Algorithm Based on Ceph Storage System

CHEN Fa-He, CHAI Xiao-Li

(The 32nd Research Institute, China Electronics Technology Group Corporation, Shanghai 201808, China)

Abstract: The Ceph storage system suffers from the metadata server performance bottleneck and low file reading efficiency during small file storage. In view of this, the study takes advantage of the inherent data relevance between small files to extract related features with a lightweight pattern matching algorithm. On this basis, small files are merged, as a result of which the rationality of file merging is improved. The small files in the same merged file are stored in the client cache to improve the cache read hit rate. Experiments have verified that the proposed scheme can effectively enhance the access efficiency of small files.

Key words: distributed storage; small file storage; data relevance; file merging

随着通信技术和超级计算技术的不断发展, 越来越多的数据出现在我们生活中, 因此高效的存储技术成为当今科学技术发展的重点. 目前互联网各行各业迅速发展, 社交媒体、教育医疗、电子商务等行业产生了巨大的小文件数据; 根据资料统计, 社交媒体软件中, 微博存储系统中每天保存着超过 100 亿张照片形式的小文件; 作业帮存储系统中保存的文档、课件、日志等小文件超过 500 TB; 腾讯公司的社交 APP 每天能产生超过 100 TB 的数据文件信息. 这些互联网企业保存的海量文件中, 绝大多数以浏览日志、缩略图为主[1–4]. 太平洋西北国家实验室研究统计表明[5], 在这些存储系统中, 超过一半的文件其大小不超过 64 KB. 传统的存储技术如磁盘阵列技术已经难以满足众多数据的存储需求. 集群存储的扩展性被越来越看重并得到了广泛的发展和应用, 其中就包括分布式文件系统.

分布式文件系统在管理本地存储节点的同时, 还可以通过网络管理其他节点, 具有优秀的扩展性, 成为了存储系统发展的重中之重. 针对海量小文件的存储的应用也已经有了一些研究, 例如国外 Facebook 的 HaystackFileSystem 存储系统和国内阿里巴巴的 TFS (TaobaoFileSystem) 等都是根据自身小文件特点设计的分布式文件系统, 其设计思路具有一定的参考意义,

① 收稿时间: 2021-04-19; 修改时间: 2021-05-19; 采用时间: 2021-06-02; csa 在线出版时间: 2022-01-17

有效的解决了因小文件数量众多导致的元数据瓶颈问题, 但是在面临高并发访问的状态下, 仍然存在存储效率较低的现象, 并且还存在未考虑小文件之间的关联性导致小文件分散分布、各自针对自身小文件特点不具有通用性、维护成本高等问题[6–9], 因此还具有很大的优化空间.

1 相关工作

1.1 研究现状

针对海量小文件存储问题, 国内外众多学者进行了很多深入研究, 总体分为两种思路: 一种是将小文件按照某种规则合并成大文件进行管理; 另一种是增加缓存、预取操作来减少相应的小文件访问网络开销[10,11]. 李孟等人[12] 提出了一种关于应对教育资源小文件的存储方式, 该方案介绍了基于关联文件的缓存、预取、以及索引等提高系统访问性能的操作, 虽然提高了教育资源类小文件的访问性能, 但是仅局限于教育资源类文件的连续性上, 如课件之间的逻辑顺序等. 赵跃龙等人[13] 通过实验分析, 发现分布式存储系统在处理小文件时, 会产生磁盘空间利用率低、文件传输延迟的现象, 于是他们将逻辑上连续的文件数据按照顺序放在磁盘空间中, 并且将对元数据的管理放在缓存中以缓解柱结构内存不足, 通过实验验证该方案提高了对小文件的处理性能. 针对小文件访问性能问题, Dong 等人[14] 将小文件分为逻辑相关文件、结构相关文件和独立文件几种类别. 然后根据不同类别的小文件采取不同的存储方案, 例如将逻辑相关的小文件采用分组预取机制, 提前预取与目标文件逻辑相关的其他小文件. 将结构相关的小文件采用文件合并的方法处理等, 经过实验对比, 这种处理方式提升了存储系统的访问性能.

1.2 分布式文件系统 Ceph

开源分布式文件系统 Ceph 的核心是高扩展、高可靠的对象存储系统 RADOS 和 CRUSH 算法, 前者为应用程序提供了一致性保证的逻辑对象存储接口; 后者高效均衡的将数据映射到存储集群中. 另外 Ceph 采用动态元数据管理方法、将元数据的管理也放到服务器中组成元数据服务器, 提高了对元数据的处理能力, 尽管 Ceph 本身具有数据双倍写入的特点, 其写操作要先写入日志再写入本地文件系统, 在大量文件访问时存在吞吐量低的问题[15–18]. 但因其优秀的可用性、可靠性以及文件接口方面的优秀设计, 所以选择用 Ceph 存储系统来实现本文的小文件合并算法优化方案.

1.3 文件关联性分析

在分布式存取系统中, 分析数据之间的关联性是非常有必要的, 例如在文件缓存和文件预取等方面. 现有的关于文件之间关联性的分析分为基于文件存取关系的关联性和基于文件数据结构的关联性.

基于文件存取关系的关联性表示为在存储系统中, 一些文件的访问常常会伴随着另一些文件同时访问, 因此在文件访问的历史队列中可以统计出一些常被一起访问的文件, 那么可以推断出它们之间依然有很高的概率被一同访问. 这种基于文件存取关系的关联性被频繁的用于文件预取等方面, 但是这种方法存在局限性, 一方面, 这种文件的关联性是基于对历史访问文件的数据分析, 因此会忽略不经常被访问文件之间的关联性; 另一方面, 在存储系统中, 若文件发生更改或者更新, 仍然根据文件更改之前的存取记录分析其关联性则失去了准确性, 因此这种方法无法准确分析出文件更新后的关联性.

基于文件数据结构的关联性是根据文件本身数据结构分析其中的关联性, 例如对于文件 A 和文件 B, 如果文件 A 中的数据内容包含对文件 B 的调用或者有指向文件 B 的链接, 那么说明文件 A 和文件 B 在本身数据结构上具有关联性. 类似的情况有很多, 例如加载同一个网页的脚本文件和图片文件; 代码编译过程中的头文件、库函数等, 根据调查结果显示[1], 相比于基于文件存取关系的关联性, 这种基于文件自身数据的关联性更普遍地存在于众多应用场景中.

在文件合并时如果能考虑到文件之间的关联性, 将有关联度的文件尽量合并在同一个文件中, 这样在文件读取时会获得更高的访问效率[19].

2 基于文件数据结构关联性的小文件合并方案

2.1 优化思路分析

在 Ceph 分布式系统中, 访问目标文件时首先通过元数据服务器获取目标文件的元数据信息, 客户端再根据获取的元数据信息到对象存储设备中获取数据文件. 访问总用时设为T, 分为处理延时T_{OP}和网络开销延迟T_{net}:

$$T = T_{OP} + T_{net} \tag{1}$$

其中, 处理延时T_{OP}分为客户端处理延时T_{client}、元数据服务器处理延时T_{mds}和对象存储设备处理延时T_{osd}.

$$T_{OP} = T_{client} + T_{mds} + T_{osd} \tag{2}$$

网络传输延迟分为客户端与元数据服务器之间、客户端与对象存储设备之间和元数据服务器与对象存储之间, 分别记为T_{climds}、T_{cliosd}、T_{mdsosd}.

$$T_{net} = T_{client} + T_{cliosd} + T_{mdsosd} \tag{3}$$

相比于大文件, 访问相同总量大小的小文件需要更多的元数据服务器访问开销T_{climds}, 因此采用小文件合并的方案仅仅在客户端处理延时T_{client}中增加了文件合并时间, 远小于客户端与元数据服务器之间的网络开销时间T_{climds}[20], 且访问文件总量越多, 优化效果更明显.

在访问总用时T中, 网络传输延迟T_{net}占据绝多大部分, 增加缓存空间和提高缓存命中率是减少网络传输延迟的有效方法, 若能将文件合并和提高缓存命中率相结合, 则可以大大提升文件访问效率.

通过上述访问用时分析, 本文提出一种新的文件合并方案: 在小文件合并时参照文件之间的数据结构关联性, 将相关联的小文件合并到一起以提高合并文件内部数据的结构合理性, 在目标小文件被访问时, 将对应合并文件缓存到客户端, 以提高缓存命中率, 减少网络开销延迟T_{net}.

2.2 文件数据关联性提取

文件数据结构间的关联性大都是通过链接的形式引出相关联的文件, 具体表现为文件通常利用关键字组成关联数据之间的引用桥梁, 因此可以利用语法分析的方法提取出有价值的关键字, 例如教育文件中某一章节的关键字如“矩阵”“行列式”; 代码文件中的“include”“import”. 通过语法分析中的模式匹配算法, 在文件中找到目标格式的数据内容即可确立文件之间的关联性, 根据用户给定的关键字和目标匹配格式就可以快速完成关联文件的提取.

对小文件关联性提取之后, 以键值对的形式来保存其中的关联性, 如图 1 所示, 系统提取两个文件具有关联性, 记之为关联文件和被关联文件, 关联顺序由关联文件指向被关联文件. 则被关联文件的元数据信息结构 (inode) 的编号存储在 8 字节的键值段中, 作为唯一标识关联文件. 另外用 score 来描述两个文件之间的关联亲密度, 其取值范围为 [0, 1], 通过实验分析得, 将关联亲密度 score 大于 0.7 的小文件进行合并时, 最终的访问效果最佳. 用 length 表示关联文件之间的路径长度, 用 offset 表示被关联文件的关联格式在关联文件中的偏移量, 将这 3 个属性分别记录在键值对中 4、4、8 个字节长度的空间中. 通过记录关联式在关联文件中的 length 和 offset 可以获得被关联文件的具体位置, 从而实现快速索引.

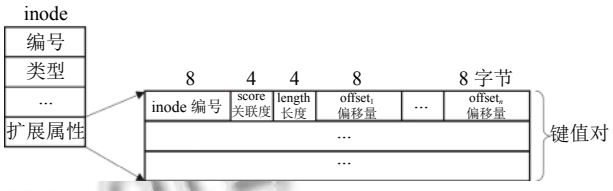

图 1 数据关联性的存储方式

2.3 小文件合并算法设计

文件合并是将小文件通过一定的算法合并成大文件, 从而减少小文件的数量, 减少应用程序在对小文件进行读写访问时客户端与元数据服务器之间的网络通信开销. 文件合并时, 在单数据节点的情况下, 可以依次将小文件合并到设定大小的合并文件中. 在分布式环境下由于采用的是分块存储机制, 因此在小文件合并过程中要考虑到分布式环境下的存储分块过程, 保证小文件的完整性, 避免不必要的访问延迟.

具体的, 在进行小文件合并过程中, 为确保每一个小文件完整的分配到一个存储块中, 合并时判定某存储块剩余空间是否足以存放下一个小文件, 不足则跳过该存储块将小文件放入下一个块中, 如图 2 所示. 这样可以避免文件跨块的现象, 有利于提高文件读取速率.

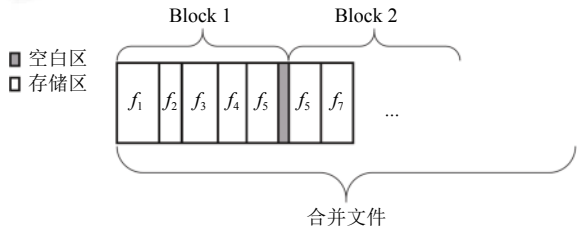

图 2 小文件合并策略

2.4 系统总架构

如图 3 所示, 本方案的主要系统架构由对象存储设备 (object storage device, OSD) 和元数据服务器 (metadata server, MDS) 和客户端 (client) 组成, 元数据服务器负责管理元数据, 内部有少量的缓存空间, 大部分元数据则存储在 OSD 中, OSD 中存储着数据文件和

元数据文件. 客户端中有文件判别模块、关联性提取模块、文件合并模块和缓存模块.

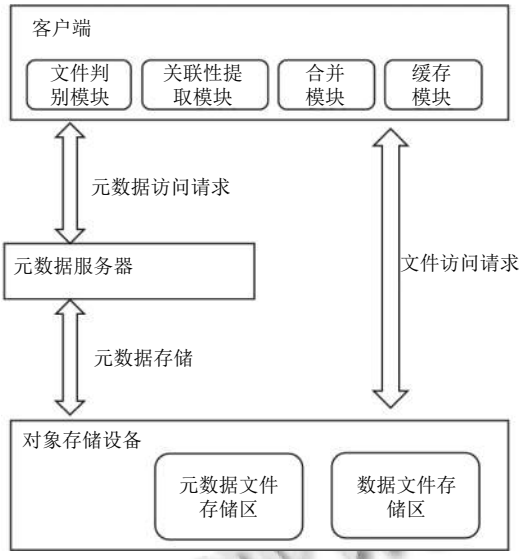

图 3 基于 Ceph 存储系统的小文件合并方案

客户端中每个模块工作原理如下:

文件判别模块: 应用程序发出一系列请求文件, 记为$D=\{F_1,F_2,F_3,F_4,\cdots,f_1,f_2,f_3,\cdots\}$首先经过文件筛选处理, 如果是大文件则直接进行存储, 如果是小文件则进入等待队列, 等待对小文件的下一步操作, 经过判别模块, 筛选出的小文件合集为$D'=\{f_1,f_2,f_3,\cdots\}$. 本文设定判别大小文件的阈值为 1 MB.

关联性提取模块: 根据第 2.2 节分析, 在小文件合并前, 参照键值对中的关联亲密度 score 值进行重排序: 以序列中第一个小文件为基础, 选中与其亲密系数 score 大于 0.7 的小文件进行排列, 剩余文件依然以余下的第一个文件为基础进行以 0.7 的亲密度分值进行排列, 得到新的小文件序列.

文件合并模块: 文件合并模块是将小文件按照新的序列进行合并, 当小文件进行合并时, 计算合并文件的大小, 保证合并文件不超过最大阈值 64 MB. 多余文件归并到下一个合并文件中. 最终形成合并文件和映射文件, 合并文件内保存小文件的数据信息, 映射文件内保存包括文件名、文件偏移量、文件逻辑块序号等小文件元数据信息.

2.5 文件写入流程

文件写入流程图如图 4, 当应用程序执行文件写入操作时, 具体步骤为: 应用程序发送目标文件集到客户端, 在客户端内首先经过文件判别, 筛选出大文件直接进行存储; 得到小文件合集进行数据结构关联性分析, 分析结果按照数据关联亲密度进行排序, 并按照排序结果执行文件合并动作, 其中每一个小文件在进行合并时判断文件大小是否小于存储块的剩余空间, 是则开始存放, 否则存放在下一个块中. 文件合并完成后生成合并文件和映射文件, 写入对象存储设备中并返回确定信息.

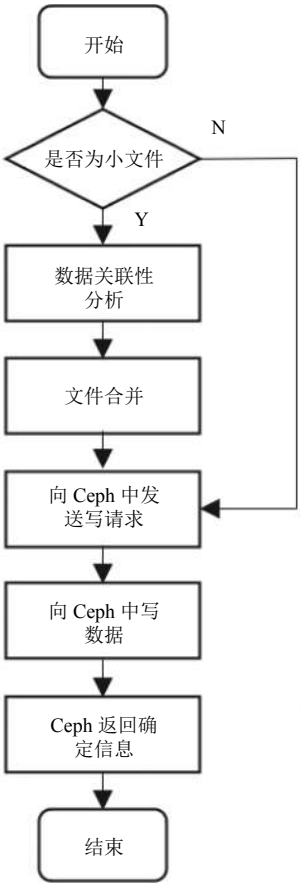

图 4 小文件合并写入过程

2.6 文件读取流程

文件读取流程如图 5 所示, 具体步骤为: 应用程序发出读文件请求传到客户端, 客户端首先判断请求文件是否为大文件, 若为大文件则直接到元数据服务器中访问目标文件元数据信息, 再到 OSD 中获取目标文件. 若为小文件, 首先确定是否存储在客户端缓存中, 若有则直接获取完成此次访问, 若无则根据元数据信息到 OSD 中获取对应的合并文件, 再根据映射文件找到合并文件中小文件的位置信息并返回到客户端, 解压合并文件获取目标文件, 合并文件中其余小文件写入缓存.

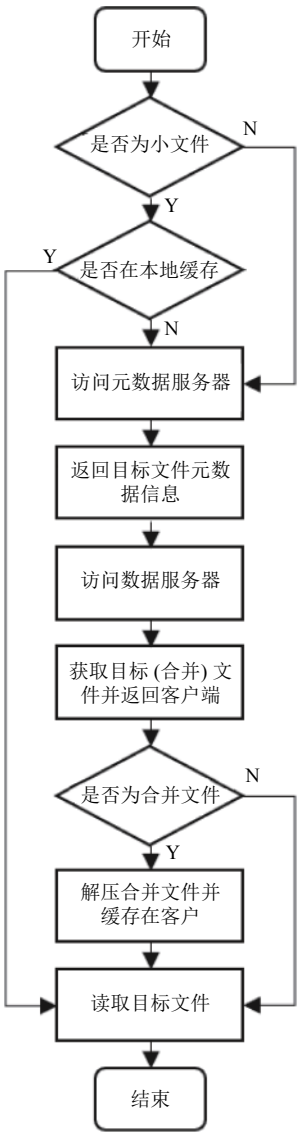

图 5 小文件读取过程

3 实验结果及分析

3.1 实验环境

根据 Ceph 分布式文件系统架构, 设计测试环境由 9 台国产申威服务器组成, 操作系统为 Ubuntu 4.18.0. 其中 8 台服务器组成集群系统, 监控节点和元数据服务器节点构建在同一台服务器上, 并设置系统为 3 副本, 文件块大小为 4 MB, 其余为存储节点; 一台服务器作为客户端, 与集群之间通过万兆交换机连接, 拓扑图如图 6.

3.2 实验设置

本文主要验证 Ceph 集群面对小文件时优化前后的读写性能指标, 为了主要测试系统面对小文件时的优化效果, 本文运用了元数据密集型的基准测试程序作为测试样本, 它使用的样本包含 12 000 个文件, 其中每个文件随机数据关联 10 个左右其他文件. 应用程序每次向客户端分别写入或读取 2 000、4 000、6 000、8 000、10 000、12 000 个小文件记录小文件平均写入和读取时间. 实验设置一组为原 Ceph 集群方案, 另一组为本文方案, 分别将两种方案下的平均写入速率和平均读取速率进行对比.

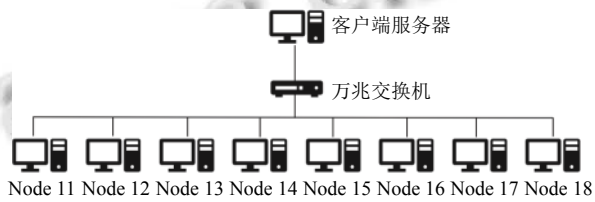

图 6 Ceph 分布式文件系统拓扑图

3.3 实验结果及分析

两组实验进行多次测试取平均值作为统计结果. 使用 IOR 作为测试工具, 图 7、图 8 为实验数据结果.

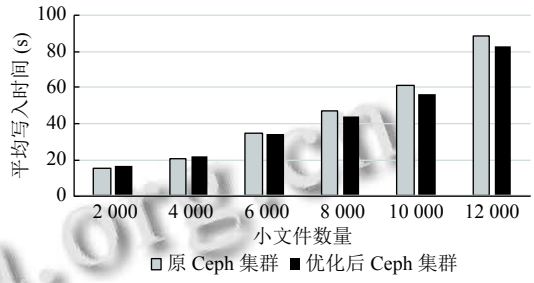

图 7 小文件平均写入时间对比

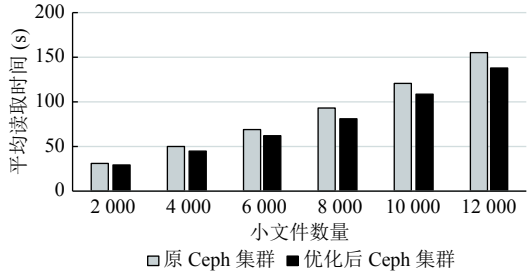

图 8 小文件平均读取时间对比

根据小文件平均写入时间实验对比结果可得, 相比于原 Ceph 集群存储方案, 本文的方案在写入较少数量的小文件时平均写入时间略高于原方案, 这是因为

本方案在客户端中增加了对小文件的处理操作如文件判别、文件关联性提取等, 因此平均写入时间略高于原方案; 文件合并后再存储可减少客户端与对象存储系统之间的网络开销, 随着写入文件数量增多, 文件合并后再存储的优势不断扩大.因此在写小文件数量超过 6 000 后, 本方案的写入平均时间更小, 在写入 12 000 个小文件时平均写入时间降低了 6.7%.

根据小文件平均读取时间实验对比可得, 本方案的文件性能明显优于原 Ceph 集群, 随着读取文件总量不断增大, 本文提出的方案在读取性能上的优势不断在增加. 在读取 2 000 个小文件和读取 12 000 个小文件时, 本方案的平均读取时间分别降低了 6.4% 和 10.9%. 这是因为本文提出的方案采用了基于数据关联性的小文件合并策略, 减少了系统访问小文件时从客户端到元数据服务器的网络开销, 并且在访问小文件时, 将与目标小文件数据关联度高的合并文件一同提取之客户端的缓存中, 有效的提高了缓存命中率, 从而减少读取和写入小文件的时间; 并且随着读取文件样本数量的不断增加, 对小文件之间关联性的分析也趋于更优状态, 因此在存取更多数量的小文件时, 本方案体现出了更好的优化效果.

4 结论与展望

本文讨论了 Ceph 分布式存储系统在面对小文件场景时存在的问题以及优化思路, 引入小文件之间的数据关联性概念并详细分析了关联性提取方法, 并将其与小文件合并方案相融合, 提高了合并文件之间的合理性. 实验结果表明, 本文提出的方案有效的提升了小文件的访问速率.

参考文献

1 陈友旭. 分布式文件系统中元数据管理优化 [博士学位论文]. 合肥: 中国科学技术大学, 2019.

2 陆小霞. 基于 Ceph 的小文件预取缓存算法. 电脑知识与技术, 2020, 16(17): 225–227.

3 Shi Q, Zhang N, Llewellyn-Jones D. Efficient autonomous signature exchange on ubiquitous networks. Journal of Network and Computer Applications, 2012, 35(6): 1793–1806. [doi: 10.1016/j.jnca.2012.07.004]

4 Yao PC. Read/write performance optimization based on Ceph heterogeneous storage. Proceedings of 2019 3rd International Conference on Computer Engineering, Information Science and Internet Technology (CII 2019). Sanya: Clausius Scientific Press, 2019. 27–37.

5 马振. Hadoop 集群中小文件的存取优化研究 [硕士学位论文]. 乌鲁木齐: 新疆大学, 2019.

6 陆小霞, 王勇, 雷晓春. Ceph 系统中海量气象小文件存取性能优化方法. 桂林电子科技大学学报, 2019, 39(1): 61–66. [doi: 10.3969/j.issn.1673-808X.2019.01.011]

7 蒋雪松. 面向小文件的分布式存储中间件的设计与实现 [硕士学位论文]. 哈尔滨: 哈尔滨工业大学, 2019.

8 樊亚. Ceph 中海量中文文本小文件存储性能优化方法研究 [硕士学位论文]. 桂林: 桂林电子科技大学, 2019.

9 Vaidya M, Deshpande S. Critical study of performance parameters on distributed file systems using MapReduce. Procedia Computer Science, 2016, 78: 224–232. [doi: 10.1016/j.procs.2016.02.037]

10 张毕涛, 辛阳. 基于 Ceph 的海量小文件存储的优化方法. 第十届中国通信学会学术年会论文集. 沈阳: 国防工业出版社, 2014. 235–240, 228.

11 Miltchev S, Smith JM, Prevelakis V, *et al*. Decentralized access control in distributed file systems. ACM Computing Surveys, 2008, 40(3): 10.

12 李孟, 曹晟, 秦志光. 基于 Hadoop 的小文件存储优化方案. 电子科技大学学报, 2016, 45(1): 141–145. [doi: 10.3969/j.issn.1001-0548.2016.01.024]

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

14 Dong B, Zheng QH, Tian F, *et al*. An optimized approach for storing and accessing small files on cloud storage. Journal of Network and Computer Applications, 2012, 35(6): 1847–1862. [doi: 10.1016/j.jnca.2012.07.009]

15 Pi Y, Wang ZL. A distributed block storage optimization mechanism based on Ceph. Proceedings of 2019 4th International Conference on Automatic Control and Mechatronic Engineering (ACME 2019). Chongqing: Clausius Scientific Press, 2019. 44–53.

16 Chen ZY, Shikh-Bahaei M. Location-aware distributed file allocation for low-delay access to electronic medical records. 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Orlando: IEEE, 2016. 5372–5375.

17 方炎柱. 分布式海量小文件存储访问优化研究与实现 [硕士学位论文]. 广州: 华南理工大学, 2019.

18 胡坚才. 面向小文件的高性能分布式文件系统研究与设计 [硕士学位论文]. 广州: 华南理工大学, 2018.

19 宋晓东. Hadoop 分布式文件系统小文件数据存储性能的优化方法研究 [硕士学位论文]. 北京: 北京交通大学, 2017.

20 葛凯凯. Ceph 文件系统元数据访问性能优化研究 [硕士学位论文]. 武汉: 华中科技大学, 2016.