大数据存储系统I/O性能优化技术研究进展

肖利民，霍志胜
北京航空航天大学计算机学院，北京 100191

摘要
大数据存储系统的I/O性能是影响大数据应用整体性能的关键因素之一，总结了当前在存储系统架构、元数据I/O性能、数据I/O性能方面开展的大数据存储系统I/O性能优化工作，并指出了未来大数据存储系统I/O性能优化的一些研究方向。

关键词
大数据存储系统；存储系统架构；元数据I/O性能；数据I/O性能；性能优化技术

中图分类号：TP399 文献标识码：A doi: 10.11959/j.issn.2096-0271.2017062

Review of I/O performance optimization technology for big data storage system

XIAO Limin, HUO Zhisheng
School of Computer Science and Engineering, Beihang University, Beijing 100191, China

Abstract
The I/O performance of big data storage system is one of the key factors that affect the overall performance of big data applications. The I/O performance optimization of big data storage system in storage system architecture, metadata I/O performance and data I/O performance was summarized, and some research directions of I/O performance optimization for big data storage systems in future were pointed out.

Key words
big data storage system, storage system architecture, metadata I/O performance, data I/O performance, performance optimization technology

1 引言

大数据已成为当前IT领域的重点研发内容和产业发展方向，我国把应对大数据问题带来的机遇和挑战提升到了国家战略层次，国家自然科学基金、国家重点研发计划等国家科技计划设置专项引导我国大数据研发工作，国务院颁布《促进大数据发展行动纲要》推动我国大数据产业发展工作。根据《自然》杂志对大数据及其应用的论述，数据存储是大数据处理和利用过程中不可或缺的关键环节。大数据存储系统是满足大数据应用存储需求的基础设施，其输入/输出（input/output，I/O）性能直接决定大数据存储效率，是影响大数据应用整体性能的关键因素。因此，如何提升大数据存储系统I/O性能是当前大数据领域的研究热点。

大数据存储系统大多沿用传统存储技术构建，甚至大多直接由传统存储系统扩展或升级而来，在大数据应用环境的巨大负载压力下，元数据I/O和数据I/O性能极易成为大数据存储过程中的性能瓶颈。例如，在大数据应用环境中，元数据I/O在整个存储系统I/O活动中占比很高，而支持元数据I/O的传统目录树结构组织方式往往是针对小规模数据设计的，不适应大数据应用导致的日益庞大的目录树规模，因此，元数据I/O极易成为影响存储系统I/O性能的关键瓶颈。同时，在大数据应用环境中，数据规模超过PB级甚至EB级，文件数量达到万亿级别，用户数量急剧增长，用户I/O负载呈现出多样性特征，且存在数据服务器中数据分布不合理、I/O带宽资源竞争剧烈、小文件大量存在等因素，数据I/O同样容易成为影响存储系统I/O性能的瓶颈。

为了适应和满足大数据应用环境中数据存储的需求，针对当前存储系统中元数据I/O和数据I/O面临的性能瓶颈问题，国内外学术界和工业界主要从存储系统架构优化、元数据I/O性能优化、数据I/O性能优化3个维度开展了大量的大数据存储系统I/O性能优化工作，如图1所示。因此，本文首先总结和分析了当前在存储系统架构优化方面的工作，包括基于负载特征的存储系统架构、密集型元数据I/O缓存架构、基于Flash的存储系统架构、新型元数据管理架构；其次，总结和分析了当前存储系统元数据I/O性能优化技术，包括元数据搜索、元数据查找、元数据创建3方面的优化技术；再次，总结和分析了当前存储系统数据I/O优化技术，包括数据I/O的文件级分条方法、数据I/O的负载均衡方法、数据I/O的最小化访问冲突方法、数据I/O的写优化技术、数据I/O的缓存容量扩展技术、数据I/O的带宽分配技术、数据I/O的客户端缓存技术；然后，分析和指出了未来大数据存储系统I/O性能优化可能的一些研究方向；最后，对全文进行了总结。

2 存储系统架构优化技术

在大数据应用环境中，当前存储系统架构主要面临如下问题。

- 大数据应用的负载特征呈现出多样性，存储系统的通用架构无法有效应对多样化的负载，从而造成存储系统I/O性能的异常起伏。
- 大数据应用使得元数据I/O请求呈现密集型负载特征，导致元数据服务器I/O带宽资源的竞争加剧，从而引起元数据I/O性能下降。
- 支持元数据I/O的传统目录树组织方式是为较小规模的存储系统设计的，不

适应大数据应用中日益庞大的目录树规模，从而限制了元数据I/O性能。

● 随着新型Flash存储介质的大量应用，亟需研究新的存储系统架构，以充分利用Flash随机读写性能优势，从而提高存储系统I/O性能。

针对上述问题，当前学术界和工业界从存储系统架构角度出发，开展了大量I/O性能优化工作。

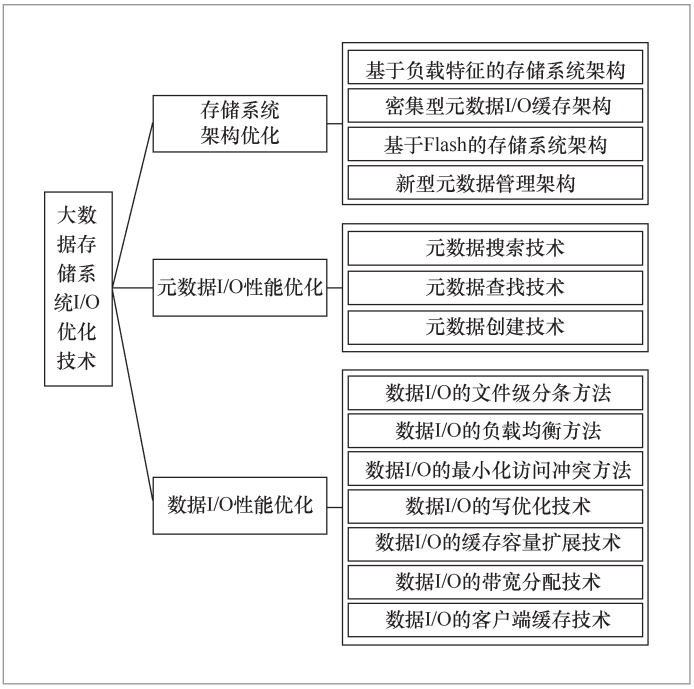

图1 大数据存储系统I/O性能优化相关研究工作

2.1 基于负载特征的存储系统架构

在大数据应用环境中，存储系统服务的I/O负载通常来自多种类型的大数据应用，普遍采用“one-size-fits-all”的存储系统架构设计，无法很好地满足大数据应用负载对存储资源访问的多样性需求，往往导致底层存储系统的性能未被充分利用。

在具有不同负载特征的并发大数据应用环境中，优化I/O负载的性能给存储系统的架构设计带来了新的需求和挑战。首先，不同类型应用对存储资源的访问具有多样性的需求，根据负载特征设计合理的I/O优化策略是满足存储需求的必要手段，并且，还需要细粒度地配置和管理启用的I/O优化策略，而目前的存储系统如并行虚拟文件系统（parallel virtual file system，PVFS）、Lustre等只支持文件数据分布等少数优化策略。其次，在系统运行过程中，针对I/O负载产生的请求，需要选择符合其特征的优化策略，以满足相应的存储访问需求，而现有的存储系统架构主要面向大规模科学计算应用而设计，难以在优化策略的实现以及I/O请求的处理过程中，区分I/O负载的数据存储和访问方式。最后，存储系统需要支持在系统不停机的情况下，根据负载特征的变化情况动态调整处理I/O请求使用的优化策略，并保证调优过程中并发访问的正确性，而现有的静态配置和动态配置方法无法同时满足上述需求。

为优化存储系统架构，使其更好地适应负载多样性，通常分别从存储系统架构的I/O高层库、I/O中间件层、存储系统层开展相应的优化，具体如下。

● 基于负载特征的I/O高层库优化：通常的思路是设计一套灵活的应用程序编程接口（application programming interface，API），以支持用户在运行时描述其复杂的I/O负载特征，如非连续内存访问结构[1]等，从而将应用负载特征传递到中间件层和存储系统层，以便后续开展针对性优化。

● 基于负载特征的I/O中间件层优化：针对大数据应用存在的请求数据量小、非连续、非对称等典型负载特征，在I/O中间件层采取列表I/O（list I/O）[2]、数据类型I/O （datatype I/O）[3]、聚集I/O[4]、聚集

缓存[5]、预取[6]、数据析取[4]、高性能便携式MPI-I/O（ROM I/O）[7]、网络文件系统用户空间远程过程调用协议库（vNFS）[8]等针对性优化方法，以提高不同I/O访问模式的应用负载的I/O性能。

● 基于负载特征的存储系统层优化：针对不同类型负载特征，动态调整和优化存储系统架构及其策略，选择合适的缓存替换策略或配置相应的存储系统优化策略[9-15]，以提高存储系统的负载适应性。

2.2 密集型元数据I/O缓存架构

在大数据应用环境中，存储系统的数据量变得十分庞大，元数据的规模也日益庞大，往往超过动态随机存取存储器（dynamic random access memory，DRAM）缓存的容量；同时，大数据应用的用户数量急剧增长，多用户并发访问造成元数据I/O呈现密集型负载特征。这些都造成了元数据I/O资源竞争加剧，影响了存储系统的元数据I/O性能。

当前互联网大数据应用环境中，用户数量的规模达到上亿级别（例如，百度、淘宝等），在高峰时期，用户并发访问存储系统带来大量的读写请求，造成元数据服务器I/O资源的竞争，从而降低存储系统的整体性能，而Flash存储介质能够有效提高存储系统的性能，因此，互联网大数据应用大量采用Flash存储介质，以提高元数据I/O的性能。

为有效应对大规模密集型元数据I/O访问需求，主要的思路是构建基于Flash的元数据缓存架构，利用Flash存储介质随机读写性能的优势，提高大规模密集型元数据I/O性能。同时，为避免密集型元数据I/O中写负载过重影响固态硬盘（solid state drive，SSD）寿命，可通过重构文件系统减少元数据写回尺寸，采用就地更新（in-place）策略降低SSD写负载[16-18]。此外，也可利用基于SSD和DRAM协作式缓存架构，在尽量延长SSD寿命的同时，提高元数据I/O性能[19]。

2.3 基于Flash的存储系统架构

当前，大部分存储系统仍采用硬盘驱动器（hard disk drive，HDD）作为主要存储介质，不能有效满足大数据应用的I/O性能需求。相比之下，Flash存储介质具有更好的随机I/O性能，能够更好地满足大数据应用的I/O性能需求。但是，如果简单地将Flash加载到存储系统中，因受传统存储架构的约束，并不能充分发挥Flash的随机读写性能。

在互联网大数据应用环境中，随着网络和硬件设备技术的发展，用户数量规模变得十分庞大，如果存储系统仍以HDD作为主要的存储介质，由于HDD固有的机械式特性，将难以满足应用的性能需求，造成极大的等待时延，降低用户的体验。Flash存储介质具有优良的随机读写性能，能够提高存储系统的性能，因此，互联网行业对基于Flash的存储架构进行了大量研究。

为使Flash在存储系统中更好地发挥其优势，满足大数据I/O性能需求，基本的优化思路是设计和构建基于Flash的存储系统架构，针对当前存储系统架构中现有协议未能充分发挥Flash随机读写性能优势的问题，研究相应的优化策略[20-23]，例如，利用SSD再使用机制预取无效数据；并行化二次写以提升写操作性能；利用追加日志机制发挥Flash的随机读写性能；设计基于新型闪存转换层（Flash translation layer，FTL）接口的Flash存储系统架构等，以发挥Flash随机读写性能优势，提高存储系统I/O性能；基于新型的SSD内部管

理机制（如强力控制器、基于奇偶校验的冗余NIC阵列（RAIN）、电容支持随机存取存储器），降低SSD的垃圾回收（garbage collection，GC）负载以提高它的性能，从而提高存储系统的I/O性能[24]。

2.4 新型的元数据管理架构

传统基于目录树的元数据管理架构已被应用三十多年，当数据规模较小时，能够有效管理存储系统的元数据。但进入大数据时代，目录树规模日益庞大，传统的目录树映射方式、元数据组织和管理方式等容易成为制约元数据I/O性能提升的瓶颈。

为提升元数据I/O性能以满足大数据应用需求，当前主要从新型目录树映射方式、基于数据库的元数据组织架构两方面开展元数据管理架构的优化。

- 新型目录树映射方式[25]：传统文件系统采用元数据与数据一对一的映射方式，不能有效利用时间负载局部性特征，限制了文件系统的性能提升，因此，设计新型的目录树映射方式，采用元数据与数据一对多的映射方法，能有效提高文件系统的元数据I/O性能，同时，利用新型的元数据完整性保护方式可提高元数据I/O的性能[26]。
- 基于数据库的元数据组织架构[27-29]：采用非结构化数据库组织和管理元数据，可有效提高元数据I/O性能，并且利用基于高速带宽的分布式数据库进行元数据管理，还可提高元数据管理架构的可扩展性。

针对大数据应用的多样性负载特征、密集型元数据负载等造成的存储系统I/O性能问题，学术界和工业界从存储系统架构角度进行了大量研究，以优化元数据I/O和数据I/O性能。由于相变存储器（phase change memory，PCM）等新型存储介质的出现，未来存储系统还可能采用更多新型存储介质以提高I/O性能。然而，简单地将这些新型存储介质添加到存储系统中，受传统存储架构的约束，往往不能有效发挥其优势。为充分发挥新型存储介质的性能，需要研究新型存储系统架构，因此，基于新型存储介质的存储系统架构仍会是当前和未来一段时间的研究热点。

3 元数据I/O性能优化技术

存储系统普遍采用目录树结构组织文件，即存储系统将文件分布到不同层次的目录中进行管理，并且按照一定的元数据分布算法进一步将文件分配到不同的元数据服务器中。这是传统存储系统采用的一种元数据组织方式，当数据规模较小时，是一种高效的元数据组织方式。然而，在大数据应用环境中，数据规模剧增，文件数量规模达到万亿级别，如果依然采用传统树型组织方式，将会导致文件管理和维护及获取文件元数据的网络交互等开销不断增大，从而造成元数据I/O性能瓶颈。为此，当前学术界和工业界在元数据I/O性能优化方面开展了大量的研究工作。

3.1 元数据搜索技术

元数据搜索是大数据存储系统中一种重要的元数据访问活动。在大数据应用环境中，由于日益增长的文件规模和传统的目录组织结构等矛盾，元数据搜索操作会产生大量的元数据访问，然而，目录树的关键路径瓶颈将使元数据服务器能力未能充分利用，从而严重制约元数据搜索性能。

在高性能计算及互联网大数据应用环境中，相关大数据应用需要及时获取数据进行处理，例如，用户需要通过搜索引擎快速浏览网页。当前，存储系统主要以目录树的形式组织网页数据，并且，大数据应用导致目录树的规模十分庞大，搜索整个目录空间会带来较长的时延。树型结构存在关键路径，关键路径造成多元数据服务器串行化搜索，降低元数据的搜索性能，因此，高性能计算和互联网行业针对元数据搜索进行了大量研究。

围绕提升元数据搜索性能的研究主要可分为以下几类。

● 基于索引结构的元数据搜索方法：该类方法通常用于桌面搜索和小规模企业文件系统搜索应用，这些应用程序包含一个一般性的关系数据库和一个倒排索引，它们部署在文件系统之外，作为一个独立的应用程序使用[30,31]，以弥补文件系统欠缺的快速搜索支持。

● 基于语义分组的元数据搜索方法：针对当前主流分层文件系统架构的限制，采用语义分组相关性技术[32,33]，通过扩展分层结构的方式，从文件中动态提取文件属性，组成基于属性语义的虚拟目录，以加速元数据搜索。

● 基于采样的元数据搜索方法：该类方法是一种快速搜索技术[34]，对目录树各分支进行采样打分，通过对分值的判定，快速剪切目录分支，缩小分层目录搜索范围，以加快元数据搜索速度。

● 基于事件通知的元数据搜索方法[35]：该类方法利用时间通知机制，替换出缓存中不活跃的文件，以提高缓存使用率。相比索引结构，可减少存储开销；相比采样方法，可提高文件搜索的准确率。

● 基于多维Bloom Filter结构的并行搜索方法[36]：该类方法利用多维Bloom Filter结构，快速缩小目录树搜索范围，并且通过MapReduce实现多元数据服务器的并行化搜索，在占用轻量负载的前提下，快速准确地获得元数据搜索结果。

3.2 元数据查找技术

元数据查找是大数据存储系统中一种常用的元数据访问活动，在元数据操作中占比很大。在大数据应用环境中，根据Lustre社区的估计[37]，单个存储系统管理的文件数量规模可达到万亿级别，当前基于目录路径的元数据查找方法会引入大量的内存占用开销，从而导致元数据查找性能瓶颈问题。

在高性能计算的大数据应用场景中，万亿级别的存储系统中目录数可达到千亿级别，文件查找把文件的路径信息映射成文件句柄，大多数的存储系统采用目录层次结构来管理文件和目录，查找一个文件需要遍历目录树，并且在访问每层目录时进行权限检查，以禁止未经授权的访问，通过引入目录路径（directory path，DP）来消除目录树的遍历。然而，基于DP的方法会引入大量的内存占用开销，导致存储系统性能的下降。因此，工业界和学术界针对元数据搜索进行了大量研究。

围绕提升元数据查找性能的研究主要可分为以下两类。

● 基于散列的元数据查找方法[38-40]：该类方法采用直接散列文件访问路径定位文件所在的元数据服务器，使用层次化的Bloom Filter矩阵标识文件所在的服务器，通过直接散列文件的名字查找对应的文件元数据，利用基于文件全路径名的散列值定位元数据服务器，可减少遍历目录路径的开销。

● 基于目录查找表的元数据查找方法[41,42]：该类方法为避免目录重命名导致的迁移开销，建立了目录路径与标识号的

一一对应关系，文件的路径查找直接通过唯一不变的标识号定位文件所在目录的元数据服务器和地址，在修改目录名时，避免迁移目录中文件的元数据，从而提高元数据查询性能。

3.3 元数据创建技术

元数据创建是大数据存储系统中一种关键的元数据访问活动，当前在大数据应用环境中，大量文件创建操作导致密集的元数据创建操作，给元数据服务器造成了很大的资源竞争，因而降低了大数据存储系统的I/O性能。

科学计算、商业计算等大数据应用是大数据存储系统的主要应用来源，其元数据操作呈现访问密集性、数据海量性等特征。例如，高性能计算机普遍采用检查点技术实现系统容错，这些应用程序一般采用多进程共享文件（多对1：N-1）或单一文件（1对1：N-N）两种方式操作，但由于在高性能计算机中有数千甚至数万个进程同时执行上述应用程序，产生大规模的并发密集型元数据访问。此外，随着数据密集型应用的快速发展，应用对计算机I/O系统需求的数据总量大、文件数量多，对存储系统的元数据访问也呈现密集型的特点，如北美电力网络平台每年产生15 TB的原始数据，为不同领域应用分析的数据量超过每天45 TB。因此，针对元数据的密集型创建，学术界和工业界进行了大量研究。

围绕有效提升元数据创建性能的研究主要可分为以下两类。

● 文件创建的交互协议优化方法[43,44]：该类方法在文件创建过程中控制客户端与服务器端的消息交互流程，通过合并子操作、出租数据文件句柄、预创建数据文件等文件创建方法，在保证文件创建过程中数据一致性的前提下，减少不必要的消息交互，以提高元数据创建的性能。

● 元数据的存储优化方法[45,46]：该类方法主要通过优化文件元数据在元数据服务器上的存储方式，提升元数据在持久存储中的写入性能，从而提高元数据创建的性能。

针对大数据存储系统中目录树组织方式造成的元数据I/O性能瓶颈问题，学术界和工业界在搜索、查找、创建等元数据I/O活动方面开展了性能优化工作，一定程度上提高了元数据I/O的性能。然而，随着大数据应用的发展，数据规模的进一步扩大对元数据I/O性能提出了更高的要求，新的应用负载的出现也给元数据I/O性能带来了更多的挑战，因此，元数据I/O性能优化仍会是未来一段时间的研究热点。

4 数据I/O性能优化技术

在大数据应用环境中，数据规模达到PB甚至EB级、文件数量达到万亿级别、用户数量急剧增长、用户I/O负载呈现多样性特征，加之数据在数据服务器中分布不合理等因素，导致数据服务器资源竞争加剧、数据I/O负载不均衡，从而造成数据I/O性能瓶颈问题。为此，当前学术界和工业界在数据I/O性能优化方面开展了大量的研究工作。

4.1 数据I/O的文件分条方法

文件分条是存储系统提高数据I/O性能的重要技术途径之一，即把一个文件分成多个子文件，将这些子文件合理地分配到不同的I/O服务器上，那么访问该文件时，就可以从这些子文件中并行读取数据，从而提高数据I/O性能。因此，恰当的

文件分条方法有助于提高数据的I/O性能，而不恰当的文件分条方法会导致数据I/O性能损失[47]。

对于高性能计算而言，大数据计算应用的计算结果的速率与计算结果存储在磁盘内的速度之间的不匹配是一个不可避免的问题[11]。作为一种可行的解决方案，存储系统（例如PVFS、Lustre、通用并行文件系统（GPFS）和Panasas文件系统（PanFS））首先把一个文件分割成多个子文件，然后把这些子文件分配到多个数据服务器，从而通过多个数据服务器的并发度提供高速的数据传输速率和吞吐量。文件分条的一个关键因素是文件分条宽度的大小。具体而言，文件分条宽度是指同一个文件内被分配到同一个数据服务器上的连续文件空间。研究表明，合适的文件分条宽度的大小能够很好地提高整个存储系统的性能，否则并行文件系统80%的性能可能损失。

围绕文件分条方法的研究主要可分为以下3类。

- 系统级文件分条方法：典型的思想是基于整个并行I/O系统的平均文件大小和平均文件请求频率等因素，通过实验或性能分析模型等确定整个文件系统内文件的分条宽度。
- 目录级文件分条方法：基本的思想是位于同一目录的文件都从父目录处获得文件分条的宽度[48]。
- 文件级文件分配方法：每一个文件可根据各自承担的负载特征以及整个系统的负载情况，决定其各自的分条宽度。此外，允许某个文件在其负载特征发生改变的情况下，对该文件重新进行分条。由于不同的应用程序往往具有不同的文件访问负载特征，因此，基于文件粒度的文件分条方法能更好地保证单一文件的访问性能[49-51]。

4.2 数据I/O的负载均衡方法

进入大数据时代，大数据存储系统面临巨量数据服务器管理而引起的挑战。为充分发挥所有数据服务器的系统性能，数据I/O负载在数据服务器之间应尽可能均衡分布[52-54]。然而，日益庞大的数据服务器规模将更容易导致存储系统中数据I/O负载失衡的问题。

对于高性能大数据计算应用的科学计算数据的管理而言，存储系统扮演着重要的角色，为了充分地发挥并行I/O系统的性能，存储系统内的数据服务器之间的负载应该符合均匀的分布，数据服务器之间的负载均衡能够消除系统的性能瓶颈，从而优化整个存储系统的平均响应时间和资源利用率。然而，不恰当的文件分条大小、不平衡的文件分配策略、不同应用程序访问的冲突以及异构的计算环境等一些因素均可能导致数据服务器之间负载的失衡。

围绕数据I/O负载均衡方法的研究主要可分为以下两类。

- 静态负载均衡：指在程序运行前，对文件数据按照负载比例分块，并映射到不同的I/O服务器，以实现I/O服务器间的负载均衡。该类方法简单有效，适用于具有稳定负载或可预测负载应用，如并行计算应用，因此，几乎所有主流的并行文件系统（包括GPFS[55]、Lustre①和并行虚拟文件系统版本2（parallel virtual file system v2，PVFS2）[56]等）大多采用静态均衡方法。
- 动态负载均衡：指在文件负载特性等信息未知的情况下，根据系统运行时的动态负载信息，实时调整负载在数据服务器间的分布，从而及时消除负载热点，优化整个系统的I/O性能。该类方法通过动态调整I/O负载在数据服务器间的分布，能够有效协调并行I/O间的访问，从而有效提高存

① http://wiki.lustre.org/

储系统的并行数据I/O性能[57-59]。

4.3 数据I/O的最小化访问冲突方法

存储系统通过分条文件在多个数据服务器间的分配，最小化大数据应用访问存储系统的平均响应时间，然而，现有方法一般通过平衡磁盘间负载或最小化单磁盘上文件大小方差等技术，优化文件请求的平均响应时间等性能指标，未考虑文件请求的磁盘I/O冲突概率问题，会导致数据I/O性能降低。

高性能大数据计算应用产生的科学数据的存储和管理已成为当今学术界和工业界面临的实际挑战。主要的原因是当前需要存储的数据的容量急剧增加，然而科学计算程序从磁盘上读取数据速率的增加仍然相当慢。因此，对于大多需要频繁进行文件读写的大数据科学计算应用来讲，文件数据的输入和输出仍然是其性能瓶颈之一。与此同时，存储系统能够通过在多个磁盘间分配文件分条后的文件提供并行的文件数据读和写，从而提供高速的文件数据传输速率。因此，基于大规模的并行磁盘设备建立的存储系统最近获得了相当多的研究关注，为了在存储系统内进行大数据的有效存储和管理，在文件被访问之前，应该在多个并行的磁盘内对文件进行有效的分配。然而，文件分配方法往往忽略了动态文件访问特性——文件请求的磁盘I/O冲突概率。

围绕数据I/O最小化访问冲突方法的研究主要可分为以下两类。

- 基于最优化理论的文件分配算法[60,61]：该类方法以最小化平均响应时间（或最优化资源利用率，或最大化系统吞吐量等）为目标，以数据服务器的最大化服务率等为限制条件，建立最优化模型求解文件最优分配方案。该类方法的优点是平台无关性和准确性，但其不足是计算复杂度较高。
- 基于启发式思想的文件分配算法[62-64]：该类方法充分利用和挖掘已知的文件访问信息，以其中某些文件访问特性（如文件访问的时间方差等）为出发点，建立近似均衡的负载均衡算法实现文件的分配。该类方法计算复杂度低，虽然性能较基于最优化理论的方法有所下降，但在实际应用场景下有较好的使用价值。

4.4 数据I/O的写优化技术

在大数据应用的I/O负载中，写操作占据很高的比例，同时，相比于读操作，写操作是一种代价昂贵的操作，无论存储介质是HDD还是SSD，写操作都会被放大，影响存储系统的I/O性能，因此，写操作性能是存储系统I/O性能的重要瓶颈。

在互联网大数据应用环境中，用户数量的急剧增加造成高并发的数据写，例如，淘宝在促销活动时期，每秒钟能够达到上百万的写操作，这会给存储系统带来极大的压力，造成存储系统I/O带宽资源的竞争，降低I/O性能，因此，学术界和工业界对写操作的性能优化进行了大量的研究。

围绕数据I/O写操作优化方法的研究主要可分为以下3类。

- 非阻塞写优化策略：主要通过消除异步预取页来达到非阻塞写操作的目的，通过非阻塞写策略，增加写操作的并行化程度，从而优化写操作I/O性能[65]。
- 基于Flash的写加速优化方法：在保留动态资源管理和高可用的前提下，通过基于Flash的写加速层，无缝合并虚拟化环境，加速写操作I/O性能[66]。同时，研究建立应用的访问负载模型，为应用划分合适的Flash存储空间，以提高写操作I/O性能[67]。

● 基于写索引结构的优化方法：通过使用未修改的操作系统（operating system，OS）内核架构，最小化对写优化造成的影响[68]。在索引结构上利用延迟绑定日志、分区和范围删除等技术，在保证其他操作性能的前提下，提高写操作I/O性能[69]。在持久内存中利用基于写优化基数树的索引结构，以提高写操作性能[70]。研究基于物理地址的垃圾收集机制能够降低对写操作的干扰[71]。

4.5 数据I/O的缓存容量扩展技术

大数据应用需要快速访问存储系统中的数据，当前主流的存储介质HDD因其机械操作极易成为性能瓶颈。随着DRAM、Flash等高速存储介质的成本下降，存储系统开始大量采用DRAM、Flash等高速存储介质来扩展存储系统缓存容量以提高性能，然而，简单地将DRAM和Flash等加入存储系统中，易造成缓存资源利用率低等问题。

围绕数据I/O缓存容量扩展技术的研究主要可分为如下两类。

● 基于DRAM的缓存容量扩展方法[72]：该类方法利用DRAM扩大缓存容量，使大数据应用的大部分工作数据都装载在缓存中，同时，研究建立基于应用的访问负载模型，利用基于日志结构的管理机制提高缓存资源利用率。

● 基于Flash的缓存容量扩展方法[73-75]：该类方法利用Flash存储作为二级缓存，扩展存储系统的缓存容量。为高性能网络连接的Flash存储介质划分虚拟地址层，达到扩展缓存容量的目的，从而提高数据I/O的性能。

4.6 数据I/O的带宽分配技术

I/O带宽是大数据存储系统的一种重要资源，I/O带宽能否公平和高效地分配，直接影响着大数据存储系统的I/O性能好坏。然而，当前存储系统的异构性和应用负载的多样性相互交织在一起，给公平和高效的I/O带宽分配带来了很大挑战。

数据中心是大数据存储的重要载体，当前互联网应用和高性能计算研究都建立了大量的数据中心。随着互联网技术的发展，数据中心的应用数量也在急剧增长。现代数据中心是为多种大数据应用服务构建的，不同的大数据应用表现出不同的资源需求和负载特征，如何分配存储系统I/O带宽资源影响着大数据应用的性能。同时，随着价格下降，SSD凭借优良的随机读写性能在存储系统中被广泛地采用，以提高I/O性能，SSD与HDD共存造成存储介质的异构性。由于不同服务器中SSD和HDD的配置不同，导致服务器间I/O带宽能力具有差异性。存储系统的异构性、服务器间带宽能力的差异性、应用间I/O负载的不同等因素相互交织在一起，给公平和高效的存储系统I/O带宽分配带来新的挑战。因此，学术界和工业界对I/O带宽分配进行了大量研究。

围绕数据I/O带宽分配的研究主要可分为如下几类。

● 基于成比例公平性的单种资源分配方法[76-79]：该类方法利用应用的负载权重，在用户间提供严格的成比例I/O带宽资源分配。该类方法在CPU资源分配、网络带宽资源分配、存储系统I/O调度和服务质量（quality of service，QoS）保证等方面已获应用。但是，该类方法仅可用于单一资源的分配，无法有效应对多种资源的分配。另外，由于成比例公平性的限制，其资源利用率也不高。

● 基于成比例公平性的多种资源分配方法[80-86]：该类方法基于应用的全局系统瓶颈资源，进行I/O带宽资源分配，所有应

用的全局系统瓶颈资源的共享比例相同。该类方法能够有效应对多种资源的分配，但由于成比例公平性属性的限制，其资源利用率不高，并且该类方法把所有服务器看成一个资源池，无法为用户确定每台服务器上的I/O带宽分配额。

● 基于不成比例公平性多种资源分配方法[87]：该类方法基于应用的局部瓶颈资源进行I/O带宽资源分配，位于同一瓶颈资源分组的所有应用对该瓶颈资源的分配比例相同。该类方法可同时分配多种资源，并且其资源利用率相当高，但无法应对多异构服务器环境，无法为用户确定每台服务器上的I/O带宽分配额。

● 多服务器环境中多种资源分配方法[88,89]：该类方法基于应用的全局或局部瓶颈资源进行I/O带宽资源分配。设置应用在单台服务器中的未知分配参数，基于公平性的限制条件，获得相应的线性规划求解模型。在多服务器环境中，可同时分配多种资源，还可确定应用在每台服务器上的I/O带宽分配额。

4.7 数据I/O的客户端缓存技术

缓存技术作为一种改善访问性能的常用方法被应用于众多领域[90-94]，客户端数据缓存是提高大数据存储系统I/O性能的有效技术之一，然而，当前客户端缓存存在资源利用率低的问题，不利于有效提升存储系统数据I/O性能。

在高性能计算环境中，大数据应用会产生大量的并发I/O访问请求。随着计算机软硬件技术的发展，客户端缓存面临着越来越密集的并发访问请求。按照目前的处理器发展速度，很快会出现具有数百个核的多核/众核处理器。因此，即使存储系统的单个节点客户端也会面临高I/O并发的问题。为了充分利用硬件的并行性，采用并行操作语言或库函数（如Pthread、消息传递接口/消息传递接口-I/O（MPI/MPI-I/O）、OpenMP）编程的多线程/多进程应用成为多个领域（如科学计算、图像处理等）的研究主流。同时，很多这类应用产生的并发、突发的I/O请求具有小于1 ms的到达间隔。

面对大量的I/O高并发访问，客户端缓存的设计面临着诸多挑战。一方面原因是I/O负载具有多样性的访问模式和不同的负载特征。例如，研究表明，很多负载具有大量小于1 MB的小文件，小文件的典型实例包括网站小图片、文本文件和科学实验生成的输出文件。另一方面，对小文件的大部分访问以只读操作为主。目前存储系统中的客户端缓存主要使用基于数据块的索引结构管理缓存数据，虽然这种方法对于缓存大文件数据简单有效，但对于高并发环境中的小文件来说性能却很差。

围绕数据I/O客户端缓存优化方面的研究主要可分为如下两类。

● 缓存数据的组织结构优化方法[95-98]：该类方法采用基于数据块索引的结构，即文件的数据被切分成相同大小的数据块放入缓存空间中，并根据缓存替换算法管理缓存中的数据，以提高缓存资源利用率。

● 缓存数据的管理优化方法[99-102]：该类方法在不影响缓存性能的前提下力图减小缓存容量，大致可有两种思路：一种是通过减少缓存块的数量以节省空间，即通过确定最少的缓存存储块个数，达到提高缓存利用率的目的；另一类是通过对缓存块内容的压缩以节省空间，即通过数据压缩技术或内容解析存储技术减少实际的数据块规模。

针对大数据存储系统中因数据量激增造成的数据I/O性能瓶颈问题，学术界和工业界在文件分条、负载均衡、I/O带宽分配、客户端缓存等方面开展了大量的性能

优化工作，以提高数据I/O性能。然而，随着大数据应用的发展，数据积累速度进一步加快、新的数据I/O负载特征不断出现、新型存储介质的发展和应用，都将给存储系统的数据I/O带来新的挑战和机遇。因此，数据I/O的性能优化仍会是未来一段时间的研究热点。

5 未来研究方向

围绕大数据存储系统I/O性能问题，学术界和工业界在存储系统架构、元数据I/O、数据I/O优化方面已有大量研究工作。但是，大数据应用和存储技术的发展将为大数据存储系统I/O性能优化带来更多新的挑战和机遇，未来可能的研究方向主要包括以下几个方面。

（1）基于DRAM的存储系统

当前大数据计算模式发展迅速，批量计算、流式计算、图计算等计算模式提高了大数据应用的计算性能，高效计算模式需要快速读写大量数据，因此，下层存储系统必须大幅提高I/O性能，以满足上层计算模式的需求。然而，现有基于HDD的存储系统对计算模式的支持已达极限。相比传统的HDD，DRAM是一种更为快速的存储介质，且其容量不断提升，成本持续下降，这为大幅提升存储系统I/O性能以有效支撑高效计算模式提供了可能。因此，基于DRAM的存储系统成为一个重要的研究方向，在此方向上当前已有不少研究工作正在开展。其中，存储系统架构及其资源管理方法直接影响DRAM资源利用率，从而影响对上层应用计算模式的高效支持。因此，基于DRAM的存储系统架构及其资源管理方法会是未来一段时期的研究重点。

（2）基于多介质的混合存储系统

新型存储介质的出现为存储系统更好地满足大数据应用的需求提供了机遇，因此，基于多介质的混合存储系统是一个重要的研究方向。其中，在多介质混合存储系统架构及其一体化管理方面，基于不同类型和不同性能的存储介质构建多介质混合存储系统，需从理论模型和实验验证两方面研究不同的混合存储策略，并建立不同混合存储策略下的存储分配模型和代价模型；在存储结构感知的数据管理方面，对于更为复杂的异构存储环境，如何实现存储介质用量的自适应动态分配，将是一个极具挑战性的问题；在大数据分布式协同存储方面，大数据应用I/O负载具有多样性和复杂性，基于新的粒度模型的热点数据识别和数据迁移将会是一个重要的问题；在高性能可扩展大数据存储结构方面，随着PCM等新型存储介质的快速发展和产品化，需研究和建立一个高性能可扩展的基于多介质的混合存储系统架构，在系统中可同时配置HDD、Flash、PCM、随机存取存储器（random access memory，RAM）等多种存储介质，并且能够发挥各种存储介质各自的优势，使得整个存储系统获得更优的整体性价比。

（3）元数据I/O性能优化

当前主流的存储系统仍采用目录树结构组织和管理元数据，在大数据应用环境中，目录树空间日益庞大，因此，元数据I/O仍会是影响存储系统性能的瓶颈之一。尽管学术界和工业界开展了大量研究以优化元数据I/O的性能，但是随着元数据规模的进一步扩大，元数据搜索、查找、创建等关键操作的性能预期会进一步降低。例如，随着目录树空间的增大，基于采样的元数据搜索方法的准确率会下降，而基于轻量级索引结构的元数据搜索方法，其存储负载和额外I/O开销会进一步加大，因此，存储系统的元数据I/O优化技术仍然会是未来的一个重要研究方向。

（4）数据I/O写性能优化

数据写I/O请求在整个I/O活动中占比较高，写操作性能易成为影响存储系统性能的关键瓶颈。尽管学术界和工业界围绕写操作性能优化开展了大量的研究工作，但随着用户数量的持续增长和数据规模的日益扩大，写操作负载会被进一步放大。一方面，由于当前存储系统的主要存储介质仍然是HDD，其低下的随机写性能严重影响写操作性能。具有良好随机读写性能的新型Flash存储介质是提高写操作性能的一个有效手段，但Flash存储介质寿命有限且价格较高。因此，需要针对不同的写负载特征，研究建立相应的写操作存储模型，在延长Flash寿命和降低开销的前提下，提高写操作性能。另一方面，当前写操作性能优化方法在提高写操作性能的同时，会降低其他I/O操作（如读操作、更新操作等）的性能，因此需要研究在保障其他I/O操作性能前提下有效提升写操作性能的优化技术。

6 结束语

大数据存储是大数据整个生命周期中不可或缺的基础环节，大数据存储系统I/O性能直接决定大数据存储效率，是影响大数据应用整体性能的关键因素。因此，针对当前存储系统中元数据I/O和数据I/O面临的性能瓶颈问题，学术界和工业界主要从存储系统架构优化、元数据I/O性能优化、数据I/O性能优化3个方面开展了大量的存储系统I/O性能优化工作，缓解了大数据存储系统I/O性能瓶颈，支撑了大数据应用的发展。然而，大数据应用的日益普及和存储技术的持续发展，为大数据存储系统I/O性能优化带来更多新的挑战和机遇，未来还需要在基于多介质的存储系统架构优化、元数据I/O性能优化、数据I/O写性能优化等方面持续开展进一步的研究工作。

参考文献:

[1] AVERY C, KENIN C, LI J, et al. High-performance techniques for parallel I/O[M]. Boca Raton: CRC Press, 2007.

[2] CHING A, CHOUDHARY A, COLOMA K, et al. Noncontiguous I/O accesses through MPI-I/O[C]//The 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, May 12-15, 2003, Tokyo, Japan. New Jersey: IEEE Press, 2003: 104-111.

[3] CHING A, CHOUDHARY A, LIAO W K, et al. Effcient structured data access in parallel file systems[C]//The IEEE International Conference on Cluster Computing and the Grid, May 12-15, 2003, Tokyo, Japan. New Jersey: IEEE Press, 2003: 326-335.

[4] THAKUR R, GROPP W, LUSK E. Data sieving and collective I/O in ROMIO[C]// The 7th Symposium on the Frontiers of Massively Parallel Computation, February 26, 1999, Annapolis, USA. New Jersey: IEEE Press, 1999: 182-189.

[5] LIAO W K, COLOMA K, CHOUDHARY A, et al. Collective caching: application-aware client-side file caching[C]//The 14th International Symposium on High Performance Distributed Computing, October 24, 2005, North Carolina, USA. New Jersey: IEEE Press, 2005: 81-90.

[6] CHEN Y, ROTH P C. Collective prefetching for parallel I/O systems[C]// The 5th Petascale Data Storage Workshop(PDSW'10), November 15, 2010, New Orleans, USA. New Jersey: IEEE Press, 2010: 1-5.

[7] THAKUR R, ROSS R, LUSK E, et al. Users guide for ROMIO: a high-performance,

portable mpi-io implementation[R]. [S.l.]: UNT Libraries Government Documents Department, 2004.

[8] CHEN M, HILDEBRAND D, NELSON H, et al. vNFS: maximizing NFS performance with compounds and vectorized I/O[C]// The 15th USENIX Conference on File and Storage Technologies, February 27-March 2, 2017, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2017: 301-314.

[9] MADHYASTHA T M, REED D A. Learning to classify parallel input/output access patterns[J]. IEEE Transactions on Parallel & Distributed Systems, 2002, 13(8): 802-813.

[10] WANG Y J, KAELI D. Profile-guided I/O partitioning[C]//The 17th Annual International Conference on Supercomputing (ICS '03), June 23-26, 2003, San Francisco, USA. New York: ACM Press, 2003: 252-260.

[11] PATRICK C M, KANDEMIR M, KARAKÖY M, et al. Cashing in on hints for better prefetching and caching in PVFS and MPI-IO[C]//The 19th ACM International Symposium on High Performance Distributed Computing, June 21-25, 2010, Chicago, USA. New York: ACM Press, 2010: 191-202.

[12] ALKISWANY S, GHARIBEH A, RIPEANU M. The case for a versatile storage system[J]. ACM SIGOPS Operating Systems Review, 2010, 44 (1): 10-14.

[13] NARAYAN S, CHANDY J A. ATTEST: attributes-based extendable storage[J]. Journal of Systems & Software, 2010, 83(4): 548-556.

[14] LI X, XIAO L, QIU M, et al. Enabling dynamic file I/O path selection at runtime for parallel file system[J]. The Journal of Supercomputing, 2014, 68(2): 996-1021.

[15] KIM S, KIM H, LEE J, et al. Enlightening the I/O path: a holistic approach for application performance[C]//The 15th USENIX Conference on File and Storage Technologies, February 27-March 2, 2017, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2017: 345-358.

[16] LU Y, SHU J, WANG W, et al. ReconFS: a reconstructable file system on flash storage[C]//The 12th USENIX Conference on File and Storage Technologies, February 17-20, 2014, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2014: 75-88.

[17] LEE E, BAHN H, NOH S H. Unioning of the buffer cache and journaling layers with non-volatile memory[C]//The 11th USENIX Conference on File and Storage Technologies, February 12-15, 2013, San Jose, USA. Berkeley: USENIX Association Berkeley, 2013: 73-80.

[18] OH Y, CHOI J, LEE D, et al. Caching less for better performance: balancing cache size and update cost of flash memory cache in hybrid storage systems[C]//The 10th USENIX Conference on File and Storage Technologies, February 14-17, 2012, San Jose, USA. Berkeley: USENIX Association Berkeley, 2012: 25.

[19] HUO Z S, XIAO L M, ZHONG Q L, et al. A metadata cooperative caching architecture based on SSD and DRAM for file systems[C]//The 15th International Conference on Algorithms and Architectures for Parallel Processing, November 18-20, 2015, Zhangjiajie, China. Berlin: Springer, 2015: 31-51.

[20] YADGAR G, YAAKOBI E, SCHUSTER A, et al. Write once, get 50% free: saving SSD erase costs using WOM codes[C]//The 13th USENIX Conference on File and Storage Technologies, February 16-19, 2015, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2015: 257-271.

[21] LEE C, SIM D, HWANG J, et al. F2FS: a new file system for flash storage[C]// The 13th USENIX Conference on File and Storage Technologies, February 16-19, 2015, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2015:

273-286.

[22] LEE S, LIU M, JUN S, et al. Application-managed flash[C]//The 14th USENIX Conference on File and Storage Technologies, February 22-25, 2016, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2016: 339-353.

[23] ARTEAGA D, CABRERA J B, XU J, et al. CloudCache: on-demand flash cache management for cloud computing[C]//The 14th USENIX Conference on File and Storage Technologies, February 22-25, 2016, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2016: 355-369.

[24] YAN S, LI H, HAO M, et al. Tiny-Tail Flash: near-perfect elimination of garbage collection tail latencies in NAND SSDs[C]// The 15th USENIX Conference on File and Storage Technologies, February 27-March 2, 2017, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2017: 15-28.

[25] ZHANG S, CATANESE H, WANG A A, et al. The composite-file file system: decoupling the one-to-one mapping of files and metadata for better performance[C]//The 14th USENIX Conference on File and Storage Technologies, February 22-25, 2016, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2016: 15-22.

[26] KUMAR H, PATEL Y, KESAVAN R, et al. High performance metadata integrity protection in the WAFL copy-on-write file system[C]// The 15th USENIX Conference on File and Storage Technologies, February 27-March 2, 2017, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2017: 197-212.

[27] JOHNSON C M, KEETON K, MORREY C B, et al. From research to practice: experiences engineering a production metadata database for a scale out file system[C]// The 12th USENIX Conference on File and Storage Technologies, February 17-20, 2014, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2014: 191-198.

[28] THOMSON A, ABADI D J. CalvinFS: consistent WAN replication and scalable metadata management for distributed file systems[C]// The 13th USENIX Conference on File and Storage Technologies, February 16-19, 2015, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2015: 1-14.

[29] NIAZI S, ISMAIL M, HARIDI S, et al. HopsFS: scaling hierarchical file system metadata using newSQL databases[C]// The 15th USENIX Conference on File and Storage Technologies (FAST'17), February 27-March 2, 2017, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2017: 89-104.

[30] LEUNG A, ADAMS I, MILLER E L. Magellan: a searchable metadata architecture for large-scale file systems: UCSC-SSRC-09-07[R]. California: University of California, 2009.

[31] LEUNG A W, SHAO M, BISSON T, et al. Spyglass: fast, scalable metadata search for large-scale storage systems[C]// The 7th Conference on File and Storage Technologies, February 24-27, 2009, San Francisco, USA. Berkeley: USENIX Association Berkeley, 2009: 153-166.

[32] GIFFORD D K, JOUVELOT P, SHELDON M A, et al. Semantic file systems[J]. Symposium on Operating Systems Principles, 1991, 25(5): 16-25.

[33] HUA Y, JIANG H, ZHU Y, et al. Smartstore: a new metadata organization paradigm with semantic-awareness for next-generation file systems[C]// The Conference on High Performance Computing Networking, Storage and Analysis, November 14-20, 2009, Portland, USA. New Jersey: IEEE Press, 2009: 1-12.

[34] HUANG H H, ZHANG N, WANG W, et al. Just-in-time analytics on large file systems[J]. IEEE Transactions on

Computers, 2012, 61(11): 1651-1664.
[35] TAKATA M, SUTOH A. Event-notification-based inactive file search for large-scale file systems[C]//APMRC, October 31-November 2, 2012, Singapore. New Jersey: IEEE Press, 2012: 1-7.
[36] HUO Z S, XIAO L M, ZHONG Q L, et al. MBFS: a parallel metadata search method based on bloomfilters using MapReduce for large-scale file systems[J]. The Journal of Supercomputing, 2016, 72(8): 3006-3032.
[37] GALEN S. 2015 Parallel file system requirements[R]. [S.l.]: Lustre Scalability Workshop, 2009.
[38] RODEH O, TEPERMAN A. zFS: a scalable distributed file system using object disks[C]//The 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies , April 7-10, 2003, San Diego, USA. New Jersey: IEEE Press, 2003: 20-27.
[39] ZHU Y, JIANG H, WANG J, et al. HBA: distributed metadata management for large cluster-based storage systems[J]. International Journal of Engineering Innovations & Research, 2008, 19: 750-763.
[40] BRANDT S A, MILLER E L, LONG D D E, et al. Effcient metadata management in large distributed file systems[C]//IEEE Symposium on Mass Storage Systems, April 7-10, 2003, San Diego, USA. New Jersey: IEEE Press, 2003: 290-298.
[41] LIU Z, ZHOU X M. A metadata management method based on directory path[J]. Journal of Software, 2007, 18(2): 236-245.
[42] LI X Q, DONG B, XIAO L M, et al. Adaptive tradeoff in metadata-based small file optimizations for a cluster file system[J]. International Journal of Numerical Analysis and Modeling, 2012, 9(2): 289-303.
[43] DEVULAPALLI A, WYCKO P. File creation strategies in a distributed metadata file system[C]//IEEE International Parallel and Distributed Processing Symposium(IEEE IPDPS 2007), March 26-30, 2007, Rome, Italy. New Jersey: IEEE Press, 2007: 1-10.
[44] YI L T, SHU J W, OU J X, et al. Cx: concurrent execution for the cross-server operations in a distributed file system[C]// 2012 IEEE International Conference on Cluster Computing, September 24-28, 2012, Beijing, China. New Jersey: IEEE Press, 2012: 99-107.
[45] STAERELING R V H V, APPUSWAMY R, MOOLENBROEK D C V, et al. Efficient, modular metadata management with loris[C]//The 6th IEEE International Conference on Networking, Architecture and Storage, July 28-30, 2011, Dalian, China. New Jersey: IEEE Press, 2011: 278-287.
[46] STENDER J, KOLBECK B, HOGQVIST M. BabuDB: fast and effcient file system metadata storage[C]//2010 International Workshop on Storage Network Architecture and Parallel I/Os, May 3, 2010, Incline Village, USA. New Jersey: IEEE Press, 2010: 51-58.
[47] CHEN P M, PATTERSON D A. Maximizing performance in a striped disk array[J]. SIGARCH Computer Architecture News, 1990, 18(2SI): 322-331.
[48] ROSS R B, CARNS P H, THAKUR R. PVFS: a parallel file system for linux clusters[C]//The 4th Annual Linux Showcase and Conference, October 10-14, 2000, Atlanta, Georgia. Berkeley: USENIX Association Berkeley, 2000: 391-430.
[49] SCHEUERMANN P, WEIKUM G, ZABBACK P. Data partitioning and load balancing in parallel disk systems[J]. The VLDB Journal, 1998, 7(1): 48-66.
[50] VASQUEZ H, LUDWIG T. Hint controlled distribution with parallel file systems[C]// Recent Advances in Parallel Virtual Machine and Message Passing Interface, September 18-21, 2005, Sorrento, Italy. Berlin: Springer, 2005: 110-118.

[51] DONG B, LI X, XIAO L, et al. A new file-specific stripe size selection method for highly concurrent data access[C]//IEEE 13th International Conference on Grid Computing, September 20-23, 2012, Beijing, China. New Jersey: IEEE Press, 2012: 22-30.

[52] LEE L. File assignment in parallel I/O systems with minimal variance of service time[J]. IEEE Transactions on Computers, 2000, 49(2): 127-140.

[53] KUNKEL J M. Towards automatic load balancing of a parallel file system with subfile based migration[D]. Heidelberg: Heidelberg University, 2007.

[54] SCHEUERMANN P, WEIKUM G, ZABBACK P. Data partitioning and load balancing in parallel disk systems[J]. The VLDB Journal, 1998, 7(1): 48-66.

[55] SCHMUCK F B, HASKIN R. GPFS: a shared-disk file system for large computing clusters[C]//Conference on File and Storage Technologies, January 28-30, 2002, Monterey, USA. Berkeley: USENIX Association Berkeley, 2002: 19.

[56] CARNS P H, III LIGON W B, ROSS R B, et al. PVFS: a parallel file system for Linux Clusters[C]//The 4th Annual Linux Showcase and Conference, October 10-14, 2000, Atlanta, USA. Berkeley: USENIX Association Berkeley, 2000: 391-430.

[57] LIU W, WU M, OU X, et al. Design of an I/O balancing file system on Web server clusters[C]// The 2000 International Workshop on Parallel Processing, August 21-24, 2000, Toronto, Canada. New Jersey: IEEE Press, 2000: 119-126.

[58] SCHROEDER B, GIBSON G A. A large-scale study of failures in high-performance computing systems[J]. International Conference on Dependable Systems & Networks, 2010, 7(4): 337-351.

[59] DONG B, LI X Q, WU Q M, et al. A dynamic and adaptive load balancing strategy for parallel file system with large-scale I/O servers[J]. Journal of Parallel and Distributed Computing, 2012, 72(10): 1254-1268.

[60] DOWDY L W, FOSTER D V. Comparative models of the file assignment problem[J]. ACM Computing Surveys, 1982, 14(2): 287-313.

[61] VERMA A, ANAND A. General store placement for response time minimization in parallel disks[J]. Journal of Parallel & Distributed Computing, 2007, 67(12): 1286-1300.

[62] LEE L W, SCHEUERMANN P, VINGRALEK R. File assignment in parallel I/O systems with minimal variance of service time[J]. IEEE Transactions on Computers, 2000, 49(2): 127-140.

[63] XIE T, SUN Y. A file assignment strategy independent of workload characteristic assumptions[J]. ACM Transactions on Storage, 2009, 5(3): 1-24.

[64] DONG B, LI X Q, XIAO L M, et al. An optimal candidate selection model for self-acting load balancing of parallel file system[J]. International Journal of High Performance Computing and Networking, 2012, 7(2): 123-128.

[65] CAMPELLO D, LOPEZ H, USECHE L, et al. Non-blocking writes to files[C]//The 13th USENIX Conference on File and Storage Technologies, February 16-19, 2015, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2015: 151-165.

[66] BHAGWAT D, PATIL M, OSTROWSKI M, et al. A practical implementation of clustered fault tolerant write acceleration in a virtualized environment[C]//The 13th USENIX Conference on File and Storage Technologies, February 16-19, 2015, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2015: 287-300.

[67] LI Q, SHI L, XUE C J, et al. Access characteristic guided read and write cost regulation for performance improvement on flash memory[C]//The 14th USENIX Conference on File and Storage Technologies, February 22-25, 2016,

Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2016: 125-132.
[68] JANNEN W, YUAN J, ZHAN Y, et al. BetrFS: a right-optimized write-optimized file system[C]//The 13th USENIX Conference on File and Storage Technologies, February 16-19, 2015, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2015: 301-315.
[69] YUAN J, ZHAN Y, JANNEN W, et al. Optimizing every operation in a write-optimized file system[C] //The 14th USENIX Conference on File and Storage Technologies, February 22-25, 2016, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2016: 1-14.
[70] LEE S K, LIM K H, SONG H, et al. WORT: write optimal radix tree for persistent memory storage systems[C]// The 15th USENIX Conference on File and Storage Technologies, February 27-March 2, 2017, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2017: 257-270.
[71] DOUGLIS F, DUGGAL A, SHILANE P, et al. The logic of physical garbage collection in deduplicating storage[C]//The 15th USENIX Conference on File and Storage Technologies, February 27-March 2, 2017, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2017: 29-44.
[72] RUMBLE S M, KEJRIWAL A, OUSTERHOUT J K, et al. Log-structured memory for DRAM-based storage[C]//The 12th Usenix Conference on File and Storage Technologies, February 17-20, 2014, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2014: 1-16.
[73] CULLY B, WIRES J, MEYER D T, et al. Strata: scalable high-performance storage on virtualized non-volatile memory[C]// The 12th USENIX Conference on File and Storage Technologies, February 17-20, 2014, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2014: 17-31.
[74] CAULFIELD A M, DE A, COBURN J, et al. Moneta: a high-performance storage array architecture for next-generation, non-volatile memories[C]// The 43rd Annual IEEE/ACM International Symposium on Microarchitecture (2010), December 4-8, 2010, Atlanta, USA. New Jersey: IEEE Press, 2010: 385-395.
[75] KIM H, SESHADRI S, DICKEY C L, et al. Evaluating phase change memory for enterprise storage systems: a study of caching and tiering approaches[J]. ACM Transactions on Storage, 2014, 10(4): 1-21.
[76] DEMERS A, KESHAV S, SHENKER S. Analysis and simulation of a fair queueing algorithm[J]. ACM SIGCOMM Computer Communication Review, 1989, 19(4): 1-12.
[77] GOYAL P, VIN H M, CHEN H. Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks[J]. ACM SIGCOMM Computer Communication Review, 1996, 26(4): 157-168.
[78] GULATI A, AHMAD I, WALDSPURGER C A. PARDA: proportional allocation of resources for distributed storage access[C]//The 7th USENIX Conference on File and Storage Technologies, February 24-27, 2009, San Francisco, USA. Berkeley: USENIX Association, 2009: 85-98.
[79] GULATI A, KUMAR C, AHMAD I, et al. BASIL: automated IO load balancing across storage devices[C]//The 8th USENIX Conference on File and Storage Technologies, February 23-26, 2010, San Jose, USA. Berkeley: USENIX Association Berkeley, 2010: 169-182.
[80] GULATI A, SHANMUGANATHAN G, ZHANG X, et al. Demand based hierarchical QoS using storage resource pools[C]//The 2012 USENIX Conference on Annual Technical Conference, June 13-15, 2012, Boston, USA. Berkeley: USENIX Association Berkeley, 2012: 1-13.
[81] GHODSI A, ZAHARIA M, HINDMAN

B, et al. Dominant resource fairness: fair allocation of multiple resource types[C]//The 8th USENIX Conference on Networked Systems Design and Implementation, March 30-April 1, 2011, Boston, USA. Berkeley: USENIX Association Berkeley, 2011: 323-336.

[82] GHODSI A, SEKAR V, ZAHARIA M, et al. Multi-resource fair queueing for packet processing[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 1-12.

[83] DOLEV D, FEITELSON D G, HALPERN J Y, et al. No justified complaints: on fair sharing of multiple resources[C]//The 3rd Innovations in Theoretical Computer Science Conference, January 8-10, 2012, Cambridge, UK. New York: ACM Press, 2012: 68-75.

[84] GUTMAN A, NISAN N. Fair allocation without trade[C]//The 11th International Conference on Autonomous Agents and Multiagent Systems, June 4-8, 2012, Valencia, Spain. New York: ACM Press, 2012: 719-728.

[85] PSOMAS C A, SCHWARTZ J. Beyond beyond dominant resource fairness: indivisible resource allocation in clusters[R]. Berkeley: Tech Report Berkeley, 2013.

[86] PARKES D C, PROCACCIA A D, SHAH N. Beyond dominant resource fairness: extensions, limitations, and indivisibilities[J]. ACM Transactions on Economics and Computation, 2015, 3(1): 3.

[87] WANG H, VARMAN P. Balancing fairness and effciency in tiered storage systems with bottleneck-aware allocation[C]//The 12th USENIX Conference on File and Storage Technologies, February 17-20, 2014, Santa Clara, USA. Berkeley: USENIX Association Berkeley, 2014: 229-242.

[88] WANG W, LIANG B, LI B. Multi-resource fair allocation in heterogeneous cloud computing systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(10): 2822-2835.

[89] HUO Z S, XIAO L M, ZHONG Q L, et al. Hybrid storage throughput allocation among multiple clients in heterogeneous data center[C]//IEEE 7th International Symposium on High Performance Computing and Communications, August 24-26, 2015, New York, USA. New Jersey: IEEE Press, 2015: 140-147.

[90] VAKALI A. Evolutionary techniques for web caching[J]. Distributed & Parallel Databases, 2002,11(1): 93-116.

[91] MADHYASTHA T M, REED D A. Learning to classify parallel input/output access patterns[J]. IEEE Transactions on Parallel & Distributed Systems, 2002, 13(8): 802-813.

[92] SETTLEMYER B W. A study of client-based caching for parallel I/O[D]. Clemson: Clemson University, 2009.

[93] VILAYANNUR M, SIVASUBRAMANIAM A, KANDEMIR M, et al. Discretionary caching for I/O on clusters[J]. Cluster Computing, 2006, 9(1): 29-44.

[94] WANG X, MALIK T, BURNS R, et al. A workload-driven unit of cache replacement for mid-tier database caching[C]//The 12th International Conference on Database Systems for Advanced Applications(DASFAA'07), April 9-12, 2007, Bangkok, Thailand. Berlin: Springer-Verlag, 2007: 374-385

[95] SETTLEMYER B W. A study of client-based caching for parallel I/O[D]. Clemson: Clemson University, 2009.

[96] VILAYANNUR M, SIVASUBRAMANIAM A, KANDEMIR M, et al. Discretionary caching for i/o on clusters[J]. Cluster Computing, 2006, 9(1): 29-44.

[97] MADHYASTHA T M, REED D A. Learning to classify parallel input/output access patterns[J]. IEEE Transactions on Parallel & Distributed Systems, 2002, 13(8): 802-813.

[98] WACHS M, ABD-EL-MALEK M,

THERESKA E, et al, Argon: performance insulation for shared storage servers[C]// The 5th USENIX Conference on File and Storage Technologies(FAST'07), February 13-16, 2007, San Jose, USA. Berkeley: USENIX Association Berkeley, 2007: 5.

[99] VILAYANNUR M, NATH P, SIVASUBRAMANIAM A. Providing tunable consistency for a parallel file store[C]//The 3rd USENIX conference on File and Storage Technologies(FAST'07)(FAST'05), December 13-16, 2005, San Francisco, USA. Berkeley: USENIX Association Berkeley, 2005: 17-30.

[100] FRASCA M, PRABHAKAR R, RAGHAVAN P, et al. Virtual I/O caching: dynamic storage cache management for concurrent workloads[C]//2011 International Conference for High Performance Computing, Networking, Storage and Analysis, December 29, 2011, Seatle, USA. New Jersey: IEEE Press, 2011: 1-11.

[101] LI M, VARKI E, BHATIA S, et al. Tap: table-based prefetching for Storage caches[C]//The 6th USENIX Conference on File and Storage Technologies, February 26-29, 2008, San Jose, USA. Berkeley: USENIX Association Berkeley, 2008: 1-16.

[102] LI X Q, DONG B, XIAO L M, et al. Small files problem in parallel file system[C]// The 2011 International Conference on Network Computing and Information Security, May 14-15, 2011, Guilin, China. New Jersey: IEEE Press, 2011: 227-234.

作者简介

肖利民(1970-),男,北京航空航天大学教授、博士生导师,计算机科学技术系主任,计算机系统结构研究所副所长。中国计算机学会大数据专家委员会委员、高性能计算专业委员会常务委员、容错计算专业委员会委员,中国电子学会云计算专家委员会委员。主要研究方向为大数据存储与文件系统、系统虚拟化与云计算、高性能计算机和服务器系统、计算机系统安全等。

霍志胜(1983-),男,北京航空航天大学计算机学院博士生,主要研究方向为大数据存储与分布式文件系统。

收稿日期: 2017-04-13