ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.10, October 2009, pp.2752−2765
doi: 10.3724/SP.J.1001.2009.03427

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种基于磁盘介质的网络存储系统缓存*

尹 洋[1,2+], 刘振军[1], 许 鲁[1]

[1](中国科学院 计算技术研究所,北京 100190)

[2](中国科学院 研究生院,北京 100049)

Cache System Based on Disk Media for Network Storage

YIN Yang[1,2+], LIU Zhen-Jun[1], XU Lu[1]

[1](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: yinyang@ict.ac.cn, http://www.nrchpc.ac.cn

Yin Y, Liu ZJ, Xu L. A cache system based on disk media for network storage. *Journal of Software*, 2009, 20(10):2752−2765. http://www.jos.org.cn/1000-9825/3427.htm

Abstract: With the dramatic increase in the scale of computing, the applications of network storage systems become wider, and the requirements for their I/O performance are also higher. Now with I/O heavy loaded, it becomes meaningful to cache data by using low-speed media in the I/O path between the client and network storage systems. In this paper, a cache system prototype D-Cache is designed and implemented based on disk media at block level for storage system. A two-level structure is adopted to manage disk cache and a corresponding cache management algorithm is provided for it at block level. The algorithm effectively solves the management problem of disk cache brought by the low-speed characteristic of disk media. By the use of bitmap, the cache management algorithm also eliminates the overhead of Copy on Write operations caused by write miss of disk cache. Experimental results show that the prototype can efficiently improve the overall performance for storage system with I/O heavy loaded.

Key words: cache management; disk media; network storage; cache algorithm

摘 要: 随着计算规模越来越大,网络存储系统应用领域越来越广泛,对网络存储系统 I/O 性能要求也越来越高.在存储系统高负载的情况下,采用低速介质在客户机和网络存储系统的 I/O 路径上作为数据缓存也变得具有实际的意义.设计并实现了一种基于磁盘介质的存储系统块一级的缓存原型 D-Cache.采用两级结构对磁盘缓存进行管理,并提出了相应的基于块一级的两级缓存管理算法.该管理算法有效地解决了因磁盘介质响应速度慢而带来的磁盘缓存管理难题,并通过位图的使用消除了磁盘缓存写 Miss 时的 Copy on Write 开销.原型系统的测试结果表明,在存储服务器高负载的情况下,缓存系统能够有效地提高系统的整体性能.

关键词: 缓存管理;磁盘介质;网络存储;缓存算法

中图法分类号: TP393 **文献标识码**: A

* Supported by the National Basic Research Program of China under Grant No.2004CB318205 (国家重点基础研究发展计划(973)); the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z402 (国家高技术研究发展计划(863))

Received 2008-01-09; Revised 2008-06-11; Accepted 2008-08-11

现今,NAS,SAN 等网络存储技术已为研究界和工业界广泛关注,利用存储集中化简化系统管理成为网络存储研究的一个热点,如 Xcat[1],OSCAR[2]等有效管理大规模 HPC 集群的研究以及 Blutopia[3],Frisbee[4],VAP[5],COD[6],SonD[7]等快速部署工作站或者服务器的研究.上述研究的核心方法是数据集中存储和管理,通过网络为客户机提供所需数据,以达到提高资源利用率、降低管理开销的目的.但是,随着客户机规模的日益扩大,集中化网络存储系统的 I/O 性能需求不断提升.如何进一步提高存储系统的可用性和可扩展性,保证在大规模、高负载环境下的存储应用是亟待解决的问题.

缓存技术一直以来都是网络存储领域解决性能问题的主要方法和研究重点.客户端的缓存系统能够提高整体系统的性能,但是当前的网络存储系统缓存大都基于内存构建,其空间相对较小,一些研究开始探索客户端本地磁盘作为缓存的方法.如 xCachefs[8],Sun 的 NFS[9],IBM 的 AFS[10],Coda[11],xFS[12],CAPFS[13]等分布式文件系统,它们在客户端使用本地磁盘作为缓存来提高系统的性能、可扩展性和可用性,从而缓解后端服务器的负载压力.但是,这些缓存系统大都针对具体文件系统,其通用性有限.在块一级实现的缓存可以透明地应用于绝大多数存储系统中,具有更强的通用性.但目前采用 Fibre Channel,iSCSI,NBD 和 AoE(ATA over ethernet)等协议的块级网络存储系统却缺乏类似的本地磁盘缓存系统.与此同时,高性能存储技术的一个发展趋势是 SAN 与 NAS 的结合,即由 SAN 提供块一级存储资源,并在文件一级实现分布式文件共享,如目前较先进的 Panasas[14],Luster[15]系统.这些因素都使得在 I/O 的访问路径上添加块一级的缓存对当前的网络存储系统具有较强的现实意义.

然而,磁盘相对于内存具有速度慢、非易失的特点,加之现代操作系统在块一级不允许对读写操作进行过久的等待,因此,传统的缓存管理方法并不能很好地满足块一级磁盘介质缓存管理的需要.针对这种情况,本文设计实现了基于磁盘介质的网络存储系统缓存 D-Cache,并提出了一种适用于块一级缓存的缓存管理算法.该算法考虑了磁盘介质的特性,结合当前主流操作系统本身的特点,在保证块一级缓存功能实现的同时提供了较好的性能.

本文第 1 节对系统进行概述.第 2 节描述 D-Cache 缓存系统设计和实现.第 3 节给出测试评价的结果,并对实验结果进行对比和分析.第 4 节介绍相关研究和结论.

1 系统概述

D-Cache 缓存系统位于客户机与远程网络存储系统的 I/O 路径之间,提供磁盘存储块一级的缓存服务.D-Cache 整合磁盘缓存空间和远程的存储设备,构建出虚拟存储设备(virtual device),向客户机提供与原远程存储设备兼容的块级存储服务.因此,对于客户机而言,D-Cache 系统与远程存储系统没有差异,其缓存服务对客户机的上层应用透明,具有较强的通用性.

D-Cache 基本架构如图 1 所示,D-Cache 既能够以虚拟存储设备的形式直接挂接在客户机上作为直接的客户端缓存使用,也能够以独立存储设备的形式在局域网(LAN)和广域网(WAN)上通过 FCP,ISCSI,NBD 等网络存储协议为客户机提供服务.当 I/O 请求发送到 D-Cache 向外提供的虚拟设备之后,D-Cache 的缓存管理层(cache management level)会根据具体的缓存管理策略,决定将 I/O 请求发送到本地的磁盘缓存或者是通过网络发送到远程存储设备中.

缓存系统在局域网或者广域网的 I/O 访问路径中存在的模式具有相当的复杂性,要考虑不同设备动态共享缓存资源的问题,这涉及到缓存资源的调度、一致性保证等复杂的问题.为了简化研究,我们在当前的工作中暂未考虑这些问题,而采用了缓存资源静态分配模式实现一对一的磁盘缓存功能.在具体实现过程中,我们直接将缓存以紧耦合的方式实现在客户机的本地系统中,其逻辑位置和在主机的 I/O 路径上没有区别,只是简化了多个客户机共享缓存资源的情况,并不影响本文中所描述的研究的效果和正确性.

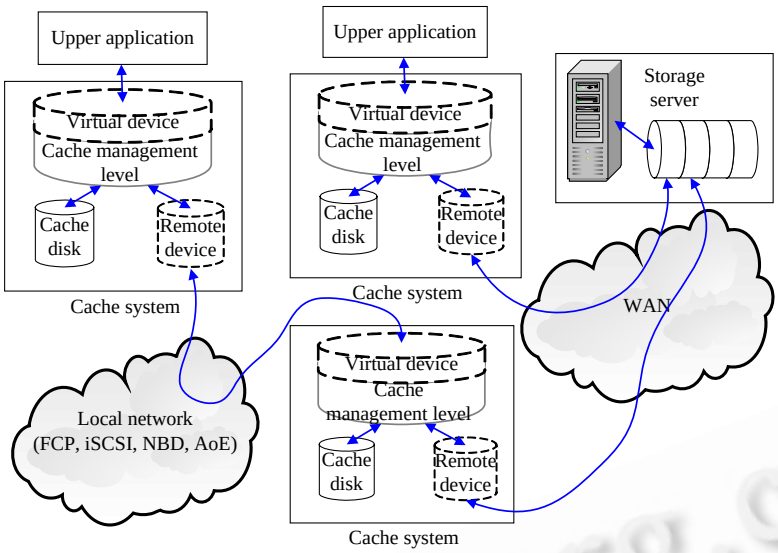


Fig.1 System architecture of D-Cache

图 1 D-Cache 系统架构

2 D-Cache 缓存系统的设计与实现

2.1 设计考虑

由于本缓存系统在介质特性和操作系统处理层次上与传统内存系统的差异较大,因此,在缓存管理和替换算法的设计上必须对这些差异点进行考虑.

2.1.1 磁盘介质特性的影响

非易失性存储:磁盘是持久性存储介质,不会因断电造成数据丢失.因此,磁盘在作为缓存的情况下,数据的回写可以根据具体的情况适当推迟.

响应速度较慢:相比于 SDRAM,flash 等缓存介质,磁盘的响应速度较慢,I/O 等待时间远高于这些高速介质.因此,缓存处理需要充分考虑这一特性,在必要时采用直接跳过缓存的方法保证系统性能.

数据连续性要求高:磁盘的连续读写性能和随机读写性能差距巨大,因此,现代的磁盘驱动器会利用本身的缓存尽量将相关联的数据放到物理上连续的位置以提高数据访问的性能.

在用磁盘等低速介质作为数据缓存的方法中,其缓存块粒度的大小对系统性能有比较明显的影响.磁盘与内存的不同之处在于,缓存块过小,比如若以一个扇区为单位,则会消耗大量的内存记录元数据,维护缓存算法的开销较高;同时,由于当前缓存替换算法并不考虑数据在缓存中的位置对性能的影响,可能导致数据在磁盘介质缓存上的不连续性.如果缓存块粒度过大,则读取数据到缓存的时间过长,为保持数据一致性,数据复制到缓存的过程中不能够对本缓存块进行正常的 I/O 操作,因此,过大粒度的缓存块将使上层应用的性能受到影响.所以,在缓存块的管理中需要充分考虑这一因素,在尽量保证缓存块数据连续性的同时,减少因 Copy on Write(COW)和缓存块替换等操作带来的 I/O 延迟等待开销.

2.1.2 操作系统块处理特性的影响

当前,主流操作系统的块一级操作和文件一级操作有所不同,文件级允许等待读写,但在操作系统的通用块一级层次上,等待时间不能过长,否则,I/O 得不到及时处理,最终将导致系统宕机.结合磁盘介质的特点,这一处理特性决定了缓存管理算法的设计中不能采用通用缓存替换算法中的替换方法(等待脏块回刷完成,再等待数据读入缓存,最后完成本次读写请求),否则会导致操作系统在通用块层进行等待的 I/O 操作过多,而致使系统宕机.

2.2 D-Cache缓存关键技术

针对上述设计考虑,D-Cache 采用了特有的两级缓存管理结构和相应的缓存管理及替换算法,针对磁盘缓存进行管理,减少了不必要的写 Miss 时的 COW 拷贝,并能够灵活配置缓存块大小,保证了缓存上数据的物理连续性,提供了较好的访问性能.

2.2.1 两级缓存管理结构

通常在缓存系统中,使用单层次的缓存块作为缓存管理的基本单元,如 Linux 操作系统使用 4K 大小的页作为基本的缓存单元.但是,磁盘的特性使得缓存块的大小不能够太大也不能够太小.若缓存块太大,则会因为预取等操作导致系统等待时间过长;若缓存块太小,则会因为在磁盘上数据分布的连续性较差而影响访问磁盘缓存的性能.D-Cache 中采用了两级层次的缓存结构来解决磁盘缓存管理中的问题.

1) 第 1 级缓存块层:缓存管理和替换的基本单位为缓存块,其大小允许用户自定义,可以根据具体应用调整缓存块大小取得较优的性能.通常,缓存块的粒度较大,在几十 KB 到几百 KB.通过大粒度的缓存块管理,不仅能够保证数据在一定范围内的连续性,还起到了数据的预取作用;
2) 第 2 级块内扇区层:在各缓存块内,将扇区作为缓存管理子单位,通过细粒度扇区级的管理,可以有效避免因写 Miss 产生缓存块级的 COW 操作.

图 2 描述了 D-Cache 缓存管理相关的元数据结构,元数据由 3 部分组成:缓存块映射表(描述缓存和源之间数据块的映射关系)、扇区脏状态位图以及扇区有效性位图.扇区的有效状态位为 1 表示该扇区的数据有效,上层应用可以直接从该缓存扇区进行读取;扇区的脏状态位为 1 表示该缓存扇区具有最新的数据,而相应源设备的对应扇区则无效.如果缓存的扇区处于脏状态,则该扇区一定是有效的.通过对各扇区的有效性进行记录,则在每次写 Miss 时不必对扇区对应的整个缓存块做 COW 操作,只需在 D-Cache 中写入该扇区新数据并将对应的脏状态位置 1 即可.由于减少了 COW 的开销,缓存系统的主要开销将集中在缓存块同步和缓存替换产生的缓存块回写操作上.

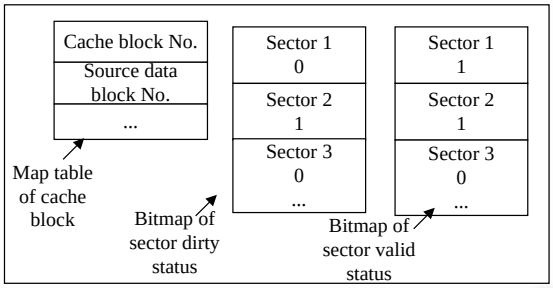


Fig.2 Metadata structure of D-Cache

图 2 D-Cache 元数据结构

虽然 D-Cache 与传统的缓存管理策略相比增加了两个位图,会额外消耗一定的内存,但与能够管理的磁盘缓存空间相比,D-Cache 位图所消耗的内存空间还是可以接受的.一个 20G 的磁盘缓存对应的位图仅需要消耗 80M 的内存资源,而一个 20G 的缓存空间对目前的绝大多数应用是足够的.

2.2.2 D-Cache 缓存管理算法

因为前述磁盘本身和操作系统的特点以及 D-Cache 采用的两级缓存管理结构,传统的缓存管理算法并不适用于 D-Cache,因此,我们设计实现了 D-Cache 缓存管理算法以适合本文所设计的磁盘介质缓存的需要.

D-Cache 缓存管理算法为缓存块定义了 7 种状态,见表 1.当缓存块处于 WRITEBACK,RESERVED 或 PART_READ 状态时,对该缓存块的读写请求将会被挂载在等待队列上进行等待,直到当前状态对应的数据操作完成后,D-Cache 再继续处理等待的读写请求.上述 7 种状态随着 I/O 访问处理的请求进行变化,通过这些状态保证缓存管理的正常进行并确保缓存块数据的有效性.

由于采用了两层缓存结构并引入了 7 种缓存块状态,D-Cache 缓存管理算法对缓存块的处理与传统管理算

法相比有很大的不同,具体如算法1所示.D-Cache在进行缓存替换时会逐个检查系统候选缓存块是否确实可以替换.如果当前缓存块处于 DIRTY,RESERVED,PART_READ,FIRSTWRITE 状态,则不能够进行替换.这时,D-Cache 会根据替换方法继续查找下一个能够替换的缓存块,直到找到一个可替换的块进行替换或者全部缓存块遍历完毕.如果 D-Cache 没有找到可以替换的缓存块,则会将遍历中遇到的第 1 个处于 DIRTY 状态的缓存块进行回刷,这时,该缓存块进入 WRITEBACK 状态并开始数据的回写.D-Cache 算法根据每个扇区的位图来判断当前缓存块中扇区是否有效,从而最终决定数据是否在缓存中命中.缓存块只是缓存管理的一个粒度,缓存命中并不表示缓存中存在有效数据,只有当表示有效状态的位图被设置为 1 时才表示对应扇区的数据是有效的.现代操作系统中,对物理存储设备写操作的基本单位是扇区,因此,通过以扇区为单位的位图信息,D-Cache 能够确认有效数据是在缓存还是在源设备中.

Table 1 States of cache block

表 1 缓存块状态

Status of cache block	Explanation
VALID	If there are some valid sectors in a cache block, then the cache block is in VALID status
INVALID	If all sectors of a cache block are not valid, then the cache block is in INVALID status
DIRTY	If there is one dirty sector in a cache block, then the cache block is in DIRTY status
WRITEBACK	When a DIRTY cache block flashes data to disk due to cache replacement operation, then the cache block is in WRITEBACK status
RESERVED	When a cache block copy data from original volume because of read miss, then the cache block is in RESERVED status during the copy process
PART_READ	If the cache hits but the target area which the read requests consists of valid sectors, the cache block will change to PART_READ status. Accordingly, the D-Cache starts to update all invalid sectors of the cache block by copying corresponding data from the source device, which guarantees all sectors of the cache block are valid after the data update
FIRSTWRITE	When an I/O write request will create new cache block entry in cache system, the new cache entry is in FIRSTWRITE status before this write request completes

算法 1. D-Cache 缓存管理算法.

```
Cache(Request){
    根据当前请求和具体的缓存替换算法在缓存中查找本请求对应的缓存块 Cache_block;
    根据查找结果和具体缓存替换算法更新缓存信息;
    if (cache_block==NULL)
        Cached_Miss(Request);
    else
        Cache_Hit(Request,Cache_block);
}
Cache_Miss(Request){
    根据当前采用的缓存替换算法选择替换的缓存块;
    if (存在可替换的缓存块 cache_block){
        if (WRITE){
            更新位图信息,设置缓存块为 VALID 和 FIRSTWRITE 状态;
            写入数据到本地缓存;
        }else{
            将缓存块设置为 RESERVED 状态;
            发起读 cache_block 操作将缓存块对应数据读取到本地缓存中,并将缓存块设置成 VALID
            状态;
            返回读请求对应的数据;
        }
    }else{
        根据替换策略选择一个脏缓存块,发起数据回写操作,并将该缓存块设置为 WRITEBACK 状态;
```

```
            将请求直接转发到远程设备;
        }
    }
    Cache_Hit(Request,cache_block){
        If (WRITE){
            If (缓存块 cache_block 处于 PART_READ 或者 RESERVED 状态){
                将请求放入等待队列中等待当前的缓存操作完成;
            }else if (cache_block 处于 WRITEBACK 状态){
                将请求放入等待队列,等待 WRITEBACK 操作完成后,直接转发 Request 到远程存储设备,
                并且将相应的缓存块状态设置为 INVALID 状态;
            }else{
                更新位图信息;
                写入数据到本地缓存;
            }
        }else{
            if (缓存块 cache_block 处于 PART_READ 或者 RESERVED 状态){
                将请求放入等待队列中,等待当前的缓存操作完成;
            }else if (请求的数据在缓存中全部有效){
                直接从缓存中读取;
            }else if (请求的数据在缓存中全部或者部分无效){
                从缓存设备和远程存储设备中分别读取所需数据;
                启动 PART_READ 操作,将本缓存块对应的其他有效数据读取到缓存中;
            }
        }
    }
```

D-Cache 的数据映射粒度不是缓存块,而是扇区.所以,D-Cache 的读写操作要根据缓存块命中情况以及表示扇区状态的位图信息来进行具体的处理.D-Cache 在写 Miss 时不用等待将原始数据读取到缓存块后再进行写操作,而可以直接写入到缓存块,并将相应扇区对应的两个位图位置 1,表示当前扇区数据脏且是有效数据.而对于读操作,因为 D-Cache 的数据映射单位是扇区,所以在读缓存块命中时并不表示缓存块中一定有读请求所需数据.缓存块命中后,D-Cache 的读操作可能存在 3 种情况,如图 3 所示:1) 读请求所需数据在缓存中全部有效;2) 读请求所需数据在缓存中部分有效;3) 读请求所需数据在缓存中不存在.在 D-Cache 算法中,我们对这 3 种情况分别处理如下:

1) 因为所有数据在缓存中全部有效,所以 D-Cache 将请求直接发送给缓存设备完成处理.
2) D-Cache 会使该缓存块进入 PART_READ 状态,对当前请求在缓存块中映射的无效扇区,D-Cache 会发起 Read Copy Job,从源设备读取相应数据返回给上层请求并写入到缓存中.对于当前请求映射到缓存块中有效扇区的部分会发起 Read Job,从缓存上读取相应的数据.而对本缓存块其他无效的扇区,D-Cache 也会发起 Read Copy Job,完成数据从源设备到缓存设备的复制以及缓存块数据的预取工作,并更新相应扇区的位图信息.为了提高读请求响应速度、减少因预读操作导致的等待开销,D-Cache 对当前读请求映射的缓存块区域进行单独处理.如图 3(b)所示,D-Cache 把当前读请求对应的读任务设置为 Main Job,其他因预读操作产生的 Read Copy Job 设置为 Sub Job.当所有 Main Job 的读操作完成后,当前的读请求即可返回,而不必等到其他 Sub Job 的完成.
3) 如图 3(c)所示,D-Cache 同样会使该缓存块进入 PART_READ 状态,然后系统发起 Read Copy Job,从源设备读取请求所需数据并写入到缓存块中.对于该缓存块中其他无效扇区的处理与上一种情况一样,也会将相应数据从源设备中复制到缓存块中.

通过上述读请求的处理,当 I/O 读请求在缓存块中没有命中或者部分命中时,会触发从源设备到缓存设备的数据复制操作.这样在复制完成后,缓存块扇区全部有效,起到了数据预取的效果,使得随后的操作可以直接在本地进行,减少了请求不命中的次数.实际测试表明,上述缓存管理算法能够有效地支撑两级缓存结构的管理,正确处理数据的请求,并取得了较好的性能.

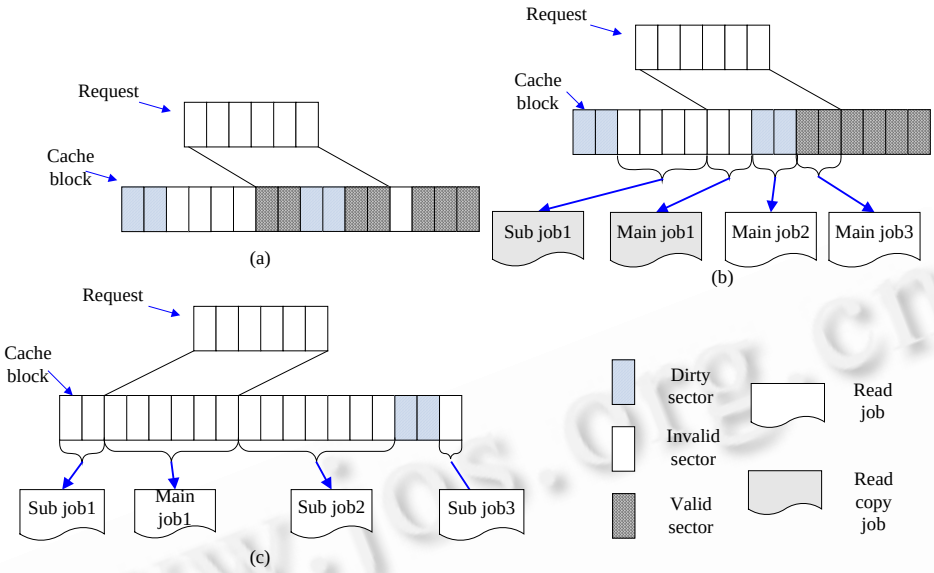


Fig.3 Different cases of the read request in D-Cache

图 3 读请求的不同情况

2.3 D-Cache缓存系统实现

我们基于 Linux 操作系统实现了 D-Cache 缓存系统原型.整个缓存系统由缓存策略管理、数据同步控制、元数据处理以及 I/O 处理、系统监控这 5 部分组成,如图 4 所示.

- 缓存策略管理是系统的核心,它负责 I/O 请求的映射,通过内部的缓存替换算法决定缓存块的替换,必要时发起缓存和源之间的数据复制请求.D-Cache 缓存策略管理模块中可以方便地集成多种缓存替换算法,比如 ARC[16],LRU,LIRS[17]等.目前,D-Cache 缓存管理算法的替换策略采用了当前主流的几种缓存替换方法,并对这些替换方法进行了针对性的简单修改.在具体的写缓存策略上,D-Cache 目前只支持 Write-Back 写方式.
- I/O 处理模块根据缓存策略完成具体的 I/O 处理,包括正常的 I/O 请求、缓存和源之间的复制操作产生的 I/O 请求以及因为读操作而产生的需要分别读写缓存和源的特殊 I/O 请求.
- 同步控制负责控制缓存与源设备之间的数据复制过程.
- 元数据处理负责完成缓存块映射表和位图信息的更新以及相关内存的申请和释放工作.
- 系统监控负责统计记录缓存系统的基本信息,包括从缓存和源进行读写的响应时间、缓存命中和不命中的情况等.

D-Cache 缓存系统各个模块按如下所述对数据 I/O 进行处理:当处于操作系统通用块层的虚拟设备接收到上层发送的 I/O 请求后,则将 I/O 请求交给 D-Cache 的缓存决策模块去处理.缓存决策模块根据当前的缓存命中情况和具体的管理策略决定是将 I/O 请求发送到等待队列中由正常 I/O 处理线程完成,还是发送到复制线程去完成,或是直接发送到物理设备中完成.如果当前缓存已满,并且所需缓存块不在缓存中,缓存决策模块会根据当前采用的缓存替换算法决定是否需要进行缓存块替换以及选择哪一个缓存块进行替换.正常 I/O 线程会从等

待队列上不断地摘除 I/O 请求发送到下层物理设备中完成具体的 I/O 操作.当数据同步事件、替换回写操作或者预读操作发生时,D-Cache 会触发复制线程,发起相应的 I/O 请求并让复制线程完成具体的复制 I/O 操作.为了防止 D-Cache 因为替换导致大量缓存块同时被回写到源设备中,造成回写负载对存储服务器的 I/O 聚集压力,我们对同时进行数据回写的缓存块数进行了限制.D-Cache 采用的具体策略是:如果当前缓存已满,且有两个缓存块正在进行回写操作,则在 I/O 请求缓存不命中时,D-Cache 不会选择一个可回写的脏缓存块发起数据回写操作,而只是将当前请求转发到存储设备上去.为了维护数据的一致性,D-Cache 中对缓存系统主动发起的同步 I/O 请求进行了特殊处理:在 D-Cache 中,正常 I/O 操作和同步 I/O 请求引发的复制操作是互斥的,同步 I/O 只有在正常 I/O 完成时才能够进行,并且选择系统空闲的时候进行数据的自动同步工作.

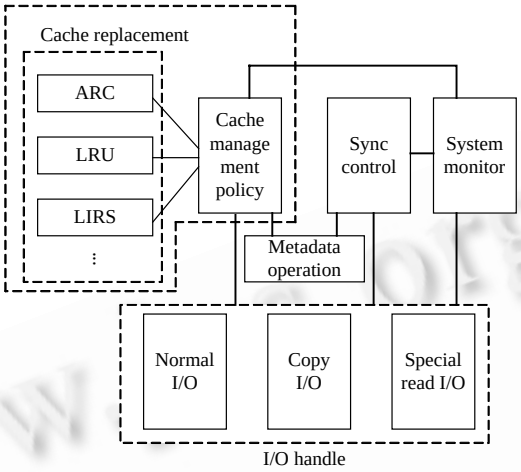


Fig.4 Logical architecture of D-Cache components

图 4 D-Cache 各部分的逻辑结构

3 评 价

3.1 测试环境

测试环境由一台存储服务器和 8 台应用服务器组成.存储服务器采用 Intel Xeon(TM) CPU 3.20GHz×2, 4GB 内存,Adaptec (formerly DPT) SmartRAID V RAID 控制器和 6 块 74GB SCSI 磁盘组成的 RAID0 存储阵列,Intel 82546GB 千兆网卡,操作系统为 32 位的 Fedora Core4,采用 2.6.11-1.1369_FC4smp 内核.每个应用服务器配置为 Intel Xeon(TM) CPU 3.20GHz×2, 4GB 内存,Adaptec (formerly DPT) SmartRAID V RAID 控制器和 3 块 74GB SCSI 磁盘组成的 RAID0 本地存储系统,Intel 82546GB 千兆网卡,操作系统为 64 位的 Fedora Core4,采用 2.6.17-1.2142_FC4smp 内核.应用服务器和存储服务器通过千兆交换机进行互联,采用 NBD(network block device)协议[18]进行数据的访问.

3.2 性能评价结果

在性能测试中,我们采用了能够方便实现分布式多机同时测试的 iozone 作为 I/O 性能的测试工具,同时选择应用服务器内存的 2 倍 8G 作为 iozone 测试文件的大小,以减少客户机内存对测试结果的影响.为了测试 D-Cache 对存储系统扩展性的影响,我们对 1 台、2 台、4 台、8 台不同数量的应用服务器分别进行了测试.在测试过程中,每个应用服务器同时启动 iozone 对存储服务器进行顺序读写并发访问.每次测试之前都会重启应用服务器,以保证应用服务器内存中没有有效数据,减少内存缓存数据对测试的影响.每次测试开始时,存储服务器导出的逻辑卷其文件系统都会被重新初始化,应用服务器也重建本地的磁盘缓存.我们对本地缓存能够缓存全部测试数据和部分测试数据的两种情况分别进行了测试,也测试了 iozone 不同读写块大小以及不同缓存

块大小对系统性能的影响.测试主要对 D-Cache 和本地磁盘、通过 NBD 直接连接的网络块设备以及 IBM 的 dm-cache[19]系统的性能作了比较**.测试中,D-Cache 和 dm-cache 都采用 Write-Back 数据写入方式.我们首先检查了 iozone 顺序测试过程中读写块大小对系统性能的影响,对 4K,32K,256K,1 024K 这 4 种不同大小的读写块分别作了测试.测试结果显示,顺序测试时,iozone 读写块的不同对性能测试结果影响不大.因此在下文中,我们主要针对 1 024K 的读写块测试结果进行讨论和分析.

3.2.1 全缓存的测试结果

我们先对本地缓存磁盘能够完全缓存 iozone 测试文件的情况进行了测试,缓存磁盘大小为 16G.我们使用两组不同性能的本地磁盘系统作为 D-Cache 在每个应用服务器上的缓存介质:3 块 72GB 的 SCSI 盘组成的 RAID5 磁盘系统和 3 块 72GB 的 SCSI 盘组成的 RAID0,缓存块大小分别为 128K 和 512K.通过图 5 所显示的测试结果我们可以发现:使用 D-Cache 缓存的存储系统能够取得更好的扩展性,其总的读写 I/O 吞吐量与直接访问本地磁盘一样,随着并发测试应用服务器数量的增加呈线性增加;而在直接使用网络设备测试时,随着并发测试服务器数量的上升,其总的读写 I/O 吞吐量没有上升,甚至下降.

图 5(a)的结果显示,当应用服务器数量从 4 台上升为 8 台时,纯网络访问测试的 read 和 reread 性能下降得十分明显,从 81M/Sec 下降到 57M/Sec,损失 20%多.这是因为,D-Cache 缓存在本地处理了绝大多数的 I/O 请求,避免了存储系统因为多个应用服务器同时大量读写造成的性能急剧下降的问题,有效地提高了系统的扩展能力.而与 dm-cache 相比,D-Cache 在第 1 次写测试的时候因为没有额外的 COW 操作的负载,新数据的写性能优势十分明显.在其他测试中,两者的整体性能差距不大,dm-cache 的 read 和 reread 的性能稍好一点,总的 I/O 带宽大概有 10M/Sec 的优势.这种性能差距是由于 D-Cache 相对复杂的缓存算法产生的开销所导致的,比如查找位图的额外开销、维护一致性的同步开销等.但总体来说,测试情况表明,D-Cache 对网络存储系统在高 I/O 负载下的可扩展性有非常显著的提高.

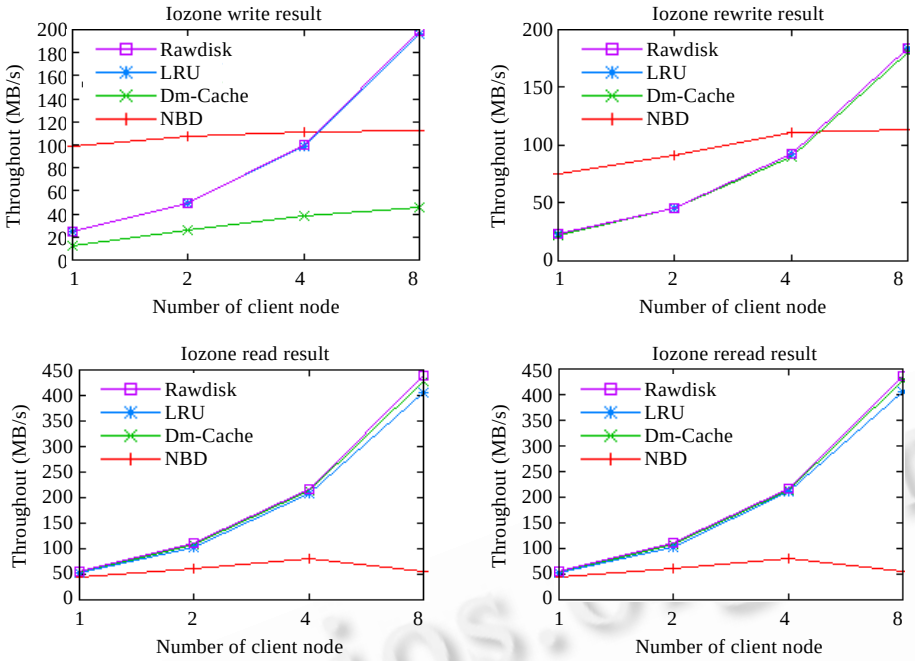
3.2.2 各种替换算法的测试结果

因为缓存容量不可能无限扩大,即使采用廉价磁盘作为存储系统的缓存,在多数应用中本地缓存系统也无法缓存全部数据.所以,为了更全面地说明 D-Cache 的效果,我们对缓存容量小于测试文件大小的情况也进行了测试.该测试中,缓存大小设置为 4G,并继续使用 iozone 对 8G 大小的文件进行顺序读写测试.因为不同的缓存替换算法测试可能会导致不同的性能,我们测试并比较了采用多种不同缓存替换算法下 D-Cache 的性能.我们在 D-Cache 框架下实现了 FIFO,LIRS[17],ARC[16],LRU,2Q[20],LFU 等 6 种算法,使得它们在保持基本替换原则的同时,又兼顾磁盘介质响应慢的特性,可以有效地在 D-Cache 框架下工作.

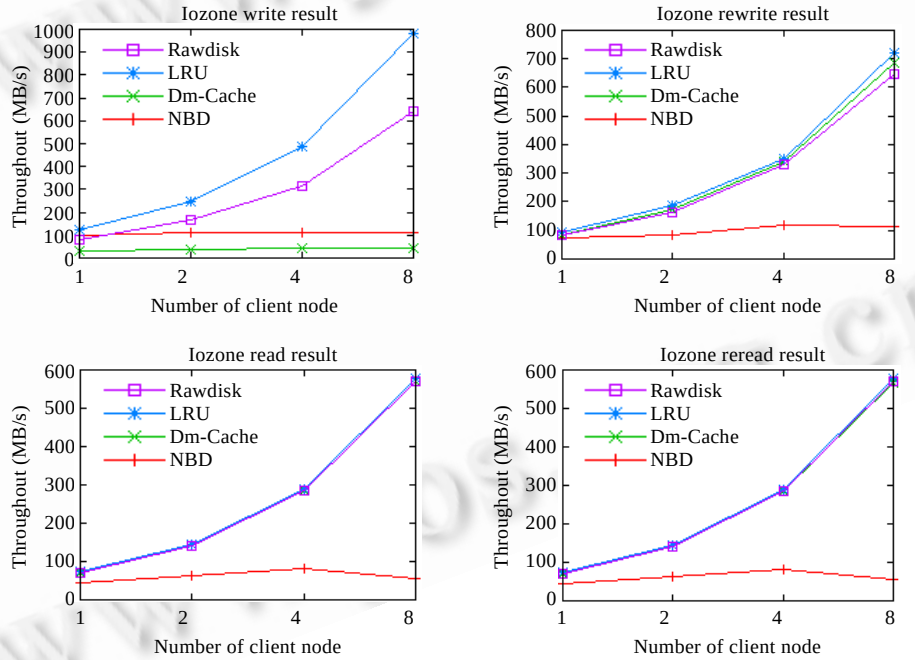
具体的测试结果如图 6 所示.从测试结果中我们可以发现,LFU 对 iozone 顺序测试具有较好的总体性能,在读写两种测试中都有较好的表现.我们通过对替换情况进行追踪后发现,在 iozone 测试中,LFU 总是替换固定的几个缓存块并将其回写,使得大部分有效数据被保存在本地缓存磁盘中,这样能够有效地提高 iozone 测试的命中率,同时又能够比较充分地利用网络带宽.所以,在应用服务器数量较少并且存储服务器 I/O 负载较低的情况下,LFU 算法也能取得相对于其他替换算法较好的 I/O 吞吐性能.

从测试中我们还可以看到,dm-cache 由于每个替换操作都需要进行 Copy on Write 操作,其读写性能受到很大的影响,甚至在 8 台应用服务器并发访问时,其性能也比网络块设备性能要低.在 LIRS 算法测试中,我们设定 HIR 块和 LIR 块缓存空间分别是缓存大小的 1/4 和 3/4.测试结果表明,该缓存替换算法虽然具有比较好的写性能,但其读性能相比于其他算法而言非常低,我们猜测,这与 HIR 块和 LIR 块所占缓存的比例有关.在 2Q 算法的测试中,我们设置 A1in,Am 和 A1out 分别为缓存大小的 25%,75%和 75%.测试结果显示,2Q 算法在第 1 次写入数据和重写数据的性能与其他算法性能相仿,但其读性能与除 LIRS 算法以外的其他算法相比下降很多.我们猜测,这也与 2Q 算法的 A1in 等参数的具体设置有关.

** dm-cache 采用 LRU 的替换算法.测试结果相关图示中,我们用 nbd 表示通过 NBD 直接连接的网络块设备,rawdisk 表示本地网络磁盘,而 D-Cache 的测试结果用其采用的具体替换策略来表示,比如 lru 表示采用 LRU 缓存替换算法的 D-Cache.



(a) Local cache media adopts RAID5 and cache block size is set to 128K
(a) 本地缓存介质采用 RAID5 方式,缓存块大小为 128K



(b) Local cache media adopts RAID0 and cache block size is set to 512K
(b) 本地缓存介质采用 RAID0 方式,缓存块大小为 512K

Fig.5 Iozone test result for 16G cache size
图 5 缓存大小为 16G 时 iozone 测试结果

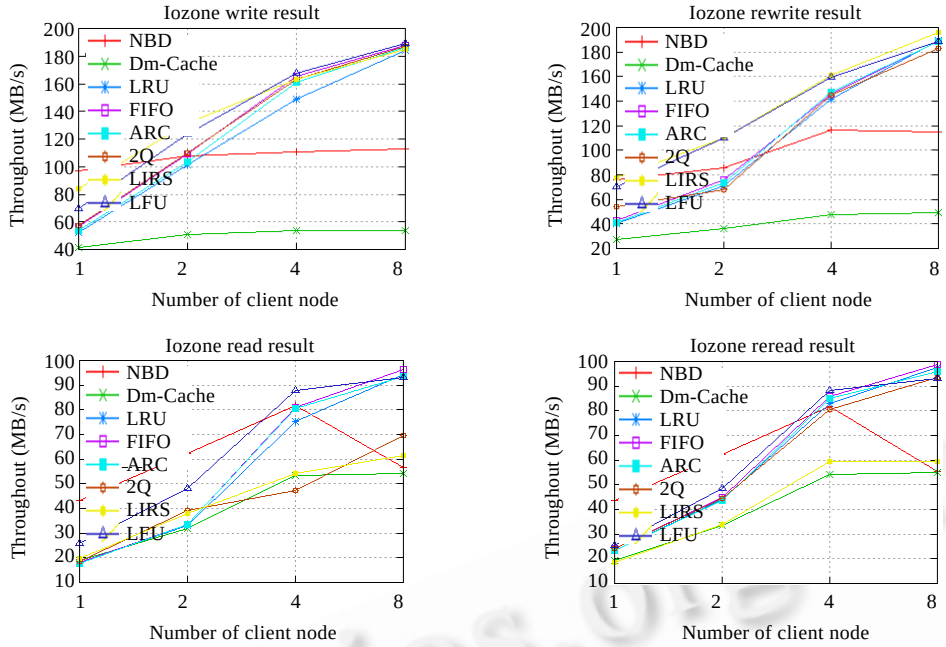


Fig.6 Iozone test result of different cache replacement algorithms with 512K cache block size and 4G cache size

图 6 4G 磁盘缓存、512K 读写块大小情况下,不同替换算法的 iozone 测试结果

3.2.3 缓存块大小对性能测试结果的影响

我们对缓存块大小对系统性能的影响也进行了相应的测试.在该项测试中,磁盘缓存大小为 4G,测试文件大小为 8G.我们主要比较了缓存块大小为 128K 和 512K 两种不同的情况,其测试结果如图 7 所示.从测试结果我们可以发现,当缓存块大小设置为 512K 时,D-Cache 取得了较高的写性能.这是因为,

1) 大的缓存块能够一次预读和回写更多数据并且使得数据在缓存设备上变得相对连续,提高了 I/O 性能;

2) 缓存块的增大减少了缓存块替换时脏缓存块数据回写的次数;

3) 大块回写时数据更加连续,在写操作异步的情况下,这也有助于提高存储服务器对请求的处理速度.

对于具有同步特性的读请求来说,缓存块虽然增大,但是同步等待从网络读取有效数据到本地缓存的时间也会增大,抵消了由于数据连续性带来的好处.因此,在应用服务器数量不多时,缓存块大小对 D-Cache 读性能的影响并不明显.而随着同时访问存储服务器的应用服务器数量的增长,大缓存块的数据连续性在读操作中的优势开始体现.因此,从测试结果中我们可以看到,在 8 台客户机时,D-Cache 的缓存块设置为 512K 较 128K 能够取得更好的读测试性能.而对于 dm-cache 来说,在写入新数据时需要先从源设备中读入整个缓存块对应的数据到本地磁盘缓存中,这样虽然缓存块变大,数据连续性得到提高,但等待读的开销也会随之增加,所以,dm-cache 的写性能在两种缓存块大小下没有较大的差异.dm-cache 的读性能在两种缓存块大小下性能差距也不大,但表现为在多客户机并发测试时,缓存块为 512K 的 dm-cache 缓存系统读性能更高.测试结果显示,D-Cache 在不同大小缓存块的情况下都能够取得比 dm-cache 更好的读性能.这是因为测试中配置的缓存大小只有测试文件的一半,顺序读写测试过程中会发生较多的替换操作,这种替换操作对两个系统的影响不一样.D-Cache 限制了同时进行回写的缓存块数量,在网络存储系统繁忙时,能够减少回写的次数,降低了存储服务器负载,提高了系统整体性能.同时,由于回写脏数据块策略的不同,使得两种方法对具体缓存块的替换和保留有所区别.这样在写后读测试时,具体缓存的数据有所差异,从而进一步导致了 D-Cache 和 dm-cache 在读性能上的差异.

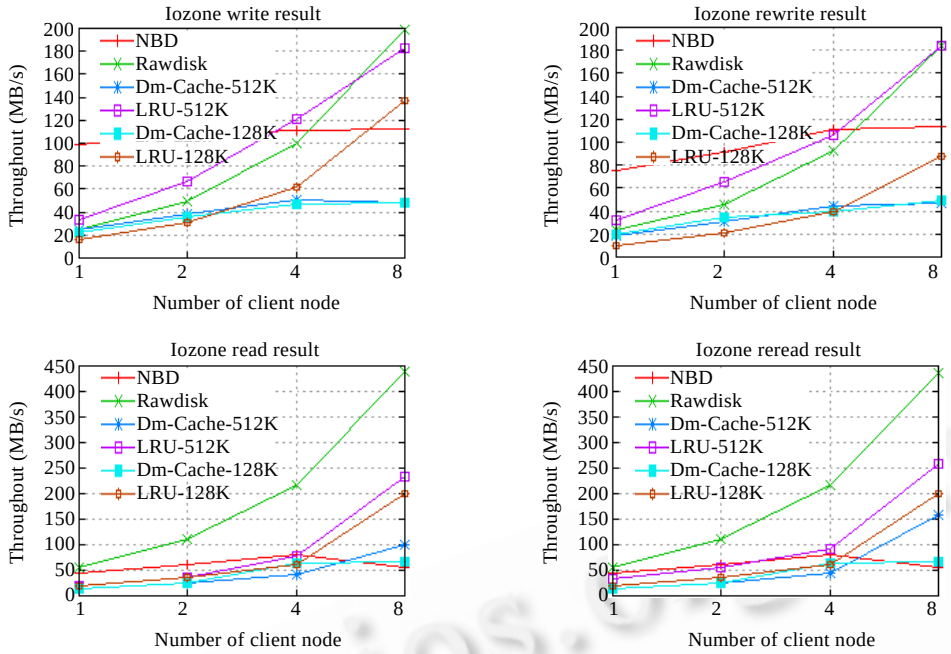


Fig.7 Comparison result of 512K and 128K cache block size

图 7 128K 和 512K 两种缓存块大小测试结果比较

4 结束语

4.1 相关研究工作

磁盘因为其廉价、非易失等特点被广泛地用作各种存储系统的缓存介质以提高数据访问的局域性,减少因网络访问拥塞和距离等原因导致的延迟,从而提高整个系统的性能.在前面我们已经提到,分布式文件系统中大量使用了磁盘作为本地的缓存系统,比如 AFS[10],xFS[12],Coda[11]等.但相比于这些在分布式文件系统中常见的文件一级磁盘介质缓存,D-Cache 具有如下优点:

1) 实现简单,不用处理各种复杂的文件系统接口,对应用透明,具有更高的通用性;

2) 采用存储网络协议,相对于分布式文件系统具有更高的性能和可扩展性.

虽然也有相关块一级缓存的研究工作在进行,比如 IBM 的 dm-cache,它和本文所描述的 D-Cache 逻辑架构基本一致,都是通过磁盘和远程的存储设备构建缓存设备,为上层应用提供数据缓存服务.但是,它与 D-Cache 的主要不同之处在于:

1) D-Cache 具有更好的整体性能,它采用两级缓存管理结构,通过位图的使用将缓存块的管理和扇区数据有效地区分对待,消除写 Miss 时的 COW 操作开销,极大提高了写操作的性能;

2) D-Cache 具有更好的扩展性,D-Cache 的框架下可以方便地实现不同缓存替换策略以适应不同应用模式,在性能上也表现出比 dm-cache 更好的扩展性.

其他的类似系统如 STICS[21]是一个通用的网络存储设备缓存系统,该系统实现了从 SCSI 协议到 IP 协议的转化,其内部的缓存结构与 DCD[22],RAIPD-Cache[23]等基本一致.该缓存系统通过本地缓存聚集小写操作能够有效地提高小写的性能,而对存储设备的数据没有起到读缓存的作用,这对于所有客户端的初始数据都来源于存储服务器的分布式应用来说帮助不大,并且相比于 D-Cache,STICS 的相关文献中并没有给出它对存储系统扩展性方面的影响.

国内研究机构清华大学利用大量内存在 iSCSI target 上实现了块一级的缓存系统 STML(SCSI target middle

level) cache[24]去提高 IP SAN 的整体性能.STML cache 利用 iSCSI Target 自带的计算处理能力和添加的大量 RAM 缓存资源来实现一个类似于 SSD 的缓存系统.STML cache 和 D-Cache 相比,首先,两者缓存介质不同导致缓存管理算法极大不同;其次,设计目标也有所不同,STML cache 主要是提高存储服务器的性能,而 D-Cache 则可以在访问存储服务器的整个 I/O 路径上向上层应用提供缓存服务,从而提高整个系统的 I/O 性能和可用性.华中科技大学研制的高可用磁盘阵列 Cache[25,26]采用 NVRAM 实现磁盘阵列的非易失缓存以提高 RAID5 的小写性能.文献[25,26]通过将写操作事务化并对映射表进行备份的方式保证缓存数据的一致性.与 D-Cache 相比,文献[25]的缓存主要面向存储服务器端的磁盘阵列而设计,采用 NVRAM 作为缓存介质,其介质特点和相应的管理方法差异较大.

在当前的热门技术——虚拟机领域也有大量的在块一级缓存的研究工作在进行.Collective[27],Machine Bank[28]都实现了基于本地磁盘的缓存.但与 D-Cache 相比,这些系统的通用性依然较差.而 D-Cache 缓存系统可以较方便地集成到不同虚拟机系统中,本地主机只需将 D-Cache 向上提供的块设备导出给虚拟机使用即可.

4.2 结论和未来工作

本文设计实现了一种基于磁盘介质的网络存储系统缓存 D-Cache,提出了适合于该缓存系统的缓存管理算法,并对该磁盘介质缓存原型系统的性能进行了评估.实验结果表明,D-Cache 能够有效地缓解大规模并发 I/O 访问时的网络存储系统负载,提高网络存储系统的可扩展性和可用性.在未来的工作中,我们会尝试提高缓存管理策略的自适应性,以求 D-Cache 能够在不同的响应速度缓存介质和数据访问负载中都取得较好的性能.当前,D-Cache 采用的同步回写策略非常简单,因此在后续工作中我们会考虑结合其他相关领域的知识,提高回写策略的智能性,并对如何提高回写效率的问题进行研究.

References:

[1] Exterme cluster administration toolkit. 2008. http://xcat.sourceforge.net/

[2] http://oscar.openclustergroup.org/

[3] Oliveira F, Patel J. Blutopia: Cluster life-cycle management. Technical Report, RC23784, IBM Research Division Austin Research Laboratory, 2005.

[4] Hibler M, Storller L, Lepreau J, Ricci R, Barb C. Fast, scalable disk imaging with Frisbee. In: Proc. of the 2003 USENIX Annual Technical Conf. San Antonio, 2003. 283−296. http://www.usenix.org/events/usenix03/tech/full_papers/hibler/hibler_html/

[5] Sapuntzakis C, Brumley D, Chandra R, Zeldovich N, Chow J, Lam MS, Rosenblum M. Virtual appliances for deploying and maintaining software. In: Proc. of the 17th Large Installation Systems Administration Conf. (LISA 2003). Berkeley, 2003. 181−194. http://www.usenix.org/event/lisa03/tech/full_papers/sapuntzakis/sapuntzakis_html/index.html

[6] Moore J, Chase J. Cluster on demand. Technical Report, CS-2002-07, Department of Computer Science, Duke University, 2002.

[7] Liu ZJ, XU L, Yin Y. Blue Whale SonD: A service-on-demand management system. Chinese Journal of Computers, 2005,28(7): 1110−1117 (in Chinese with English abstract).

[8] Sivathanu G, Zadok E. A versatile persistent caching framework for file systems. Technical Report, FSL-05-05, Stony Brook University, 2005.

[9] Sandberg R, Goldberg D, Kleiman S, Walsh D, Lyon B. Design and implementation of the Sun network file system. In: Proc. of the Summer 1985 USENIX. Portland, 1985. 119−130.

[10] Howard JH. An overview of the andrew file system. Technical Report, CMU-ITC-062, CMU Information Technology Center, 1988.

[11] Satyanarayanan M. Coda: A highly available file system for a distributed workstation environment. IEEE Trans. on Computers, 1990,39(4):447−459.

[12] Anderson T, Dahlin M, Neefe J, Patterson D, Roselli D, Wang R. Serverless network file systems. ACM SIGOPS Operating Systems Review, 1995,29(5):109−126.

[13] Vilayannur M, Nath P, Sivasubramaniam A. Providing tunable consistency for a parallel file store. In: Proc. of the 4th USENIX Conf. on File and Storage Technologies. San Francisco, 2005. 17−30. http://www.usenix.org/events/fast05/tech/full_papers/vilayannur/vilayannur_html/index.html

[14] http://www.panasas.com

[15] http://www.sun.com/software/products/lustre/index.xml

[16] Megiddo N, Modha D. ARC: A self-tuning, low overhead replacement cache. In: Proc. of the 2nd USENIX Conf. on File and Storage Technologies (FAST 2003). San Francisco, 2003. 115−130. http://www.usenix.org/events/fast03/tech/megiddo.html

[17] Jiang S, Zhang X. LIRS: An efficient low interreference recency set replacement policy to improve buffer cache performance. In: Proc. of the 2002 ACM SIGMETRICS. Marina Del Rey: ACM, 2002. 31−42.

[18] http://nbd.sourceforge.net

[19] Hensbergen EV, Zhao M. Dynamic policy disk caching for storage networking. Technical Report, RC24123, IBM Research Division Austin Research Laboratory, 2006.

[20] Johnson T, Shasha D. 2Q: A low overhead high performance buffer management replacement algorithm. In: Proc. of the 20th Int'l Conf. on VLDB. Santiago de Chile: Morgan Kaufmann Publishers, 1994. 439−450. http://www.vldb.org/conf/1994/P439.PDF

[21] He X, Zhang M, Yang Q. STICS: SCSI-to-IP cache for storage area networks. Journal of Parallel and Distributed Computing, 2004, 64(9):1069−1085.

[22] Hu Y, Yang Q. DCD—Disk caching disk: A new approach for boosting I/O performance. In: Proc. of the 23rd Annual Int'l Symp. on Computer Architecture. Pennsylvania: ACM, 1996. 169−178. http://www.ele.uri.edu/Research/hpcl/DCD/DCD.html

[23] Hu Y, Yang Q, Nightingale T. RAPID-Cache—A reliable and inexpensive write cache for disk I/O systems. In: Proc. of the 5th Int'l Symp. on High Performance Computer Architecture. Orlando, 1999. 204−213 http://www.ele.uri.edu/Research/hpcl/RAPID/RAPID.html

[24] Luo YF, Shu JW, Yu B, Wen DC. A cache system hosted on the iSCSI target in an IP SAN. In: Proc. of the 5th Int'l Conf. on Grid and Cooperative Computing-Workshops. Changsha: IEEE, 2006. 412−417. http://portal.acm.org/citation.cfm?id=1170685

[25] Xiong JG, Feng D. Design and implementation of a high availability RAID cache. Computer Engineering and Science, 2006,28(8): 119−124 (in Chinese with English abstract).

[26] Feng D, Xiong JG. Study and implementation of consistency of RAID-based NVRAM cache. Journal of Huazhong University of Science and Technology (Nature Science), 2005,33(10):70−72 (in Chinese with English abstract).

[27] Chandra R, Zeldovich N, Sapuntzakis C, Lam MS. The collective: A cache-based system management architecture. In: Proc. of the 2nd Symp. on Networked Systems Design and Implementation (NSDI 2005). Boston, 2005. 259−272. http://www.usenix.org/events/nsdi05/tech/full_papers/chandra/chandra_html/index.html

[28] Tang S, Chen Y, Zhang Z. Machine bank: Own your virtual personal computer. In: Proc. of the 21st Int'l Parallel and Distributed Processing Symp. Long Beach, 2007. http://research.microsoft.com/apps/pubs/default.aspx?id=69423

附中文参考文献:

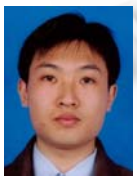
[7] 刘振军,许鲁,尹洋.蓝鲸 SonD 服务部署系统.计算机学报,2005,28(7):1110−1117.

[25] 熊建刚,冯丹.高可用的磁盘阵列 Cache 的设计和实现.计算机工程与科学,2006,28(8):119−124.

[26] 冯丹,熊建刚.磁盘阵列 cache 数据一致性的研究与实现.华中科技大学学报(自然科学版),2005,33(10):70−72.

尹洋(1980－),男,四川广安人,博士生,主要研究领域为存储系统,集群管理.

许鲁(1962－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为存储系统,体系结构,操作系统.



刘振军(1976－),男,博士,助理研究员,主要研究领域为存储系统,数据备份.