软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(4):987−995 [doi: 10.3724/SP.J.1001.2012.04046]

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

DMStone:一个分级存储系统性能测试工具*

丘建平[1], 张广艳[1+], 舒继武[1,2]

[1](清华大学 计算机科学与技术系,北京 100084)

[2](高效能服务器和存储技术国家重点实验室,山东 济南 250000)

DMStone: A Tool for Evaluating Hierarchical Storage Management Systems

QIU Jian-Ping[1], ZHANG Guang-Yan[1+], SHU Ji-Wu[1,2]

[1](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

[2](State Key Laboratory of High-End Server and Storage Technology, Ji'nan 250000, China)

+ Corresponding author: E-mail: gyzh@tsinghua.edu.cn

Qiu JP, Zhang GY, Shu JW. DMStone: A tool for evaluating hierarchical storage management systems. *Journal of Software*, 2012,23(4):987−995. http://www.jos.org.cn/1000-9825/4046.htm

Abstract: Realistic system states and representative workloads are required to evaluate a hierarchical storage management (HSM) system. The existing method to evaluate a HSM system reconstructs system states by replaying a trace within period of time. Ignoring files not used recently, the result of this method is not convincing enough. This paper proposes DMStone, a tool for evaluating HSM systems. DMStone uses file-system snapshots to generate a system state. Furthermore, it extracts workload characteristics by comparing two contiguous snapshots and generates representative workloads. DMStone can provide a realistic file-system state, including files used recently and those not used for a long time. Moreover, it makes subsequent workloads conform to realistic system states.
Key words: hierarchical storage; performance evaluation; system state; I/O workload; file system snapshot

摘 要: 对分级存储系统的性能测试,需要提供真实的系统状态和有代表性的访问负载.已有的分级存储系统测试方法通过播放一段时间的文件访问请求来生成系统状态.因为彻底忽略了近期未被访问的文件而与分级存储的真实场景不符,使得测试结果没有说服力.提出了一种分级存储系统性能测试工具 DMStone,它使用文件系统快照生成某一时刻的系统状态,并根据后续的相邻快照之间的差异提取访问负载特征,进而生成有代表性的 I/O 负载.DMStone 能够提供某一时刻真实的文件系统状态,涵盖了近期访问过的和长期不用的所有文件.而且,它能够保证后续文件访问的特征与真实应用场景相符合.
关键词: 分级存储;性能测试;系统状态;I/O 负载;文件系统快照
中图法分类号: TP316 **文献标识码**: A

对单个文件而言,在文件刚创建时访问频率最高;随着时间的推移,访问频率下降.在大规模文件系统中,大多数文件长期不用,少数文件经常使用[1].利用海量数据之间存在的访问频度差异,人们开始研制分级存储系统.它根据访问负载的变化,将具有不同访问特征的数据在具有不同性能、容量的存储设备之间动态迁移,目标是

* 基金项目: 国家自然科学基金(60903183, 61170008); 国家高技术研究发展计划(863)(2009AA01A403); 高效能服务器和存储技术国家重点实验室开放课题(2009HSSA01)

收稿时间: 2011-01-27; 修改时间: 2011-04-02; 定稿时间: 2011-04-28

在保证应用访问性能的同时,实现低成本的数据存储.

分级存储系统设计方面的研究已经发展多年[2,3],并取得较大进展,但是分级存储系统性能测试方法却依然非常滞后.性能测试的目标是:确定影响分级存储系统性能的关键因素,从而可以有针对性地进行分析和改进;比较不同的分级存储系统,为选取不同分级存储系统提供参考.分级存储系统性能测试包括两个指标:一个是将具有不同价值的数据在不同性能存储设备中分级存储的能力,另外一个是分级存储系统的 I/O 性能.分级存储系统性能测试需要通过向具有真实状态的分级存储系统播放有代表性的访问负载来完成,涉及两个主要技术挑战:创建真实的文件系统状态、生成有代表性的负载.

生成有代表性负载的问题尽管还没有彻底得到解决,但是已经朝着这个目标取得了显著进步.文件系统访问模式[4−6]和文件系统活动 trace[7,8]的实验研究已经推动了人造负载生成器[9,10]和 trace 重放方法[11,12]两方面的研究进展.相对而言,创建对于目标使用场景而言是真实的文件系统状态,也是更加困难的一个技术挑战.

目前,还没有令人满意的分级存储系统性能测试工具能够提供可用的真实的文件系统场景以及配套的文件访问记录.已有的文件系统性能测试方法往往针对空白的文件系统播放典型的负载请求,然而,分级存储系统处理的并不只是活跃的数据,而是整个文件系统内容.这种简单的文件系统测试方法因为与分级存储系统应用的真实环境不符而不适用.有人为分级存储系统提出了专用的测试方法,通过将采集到的文件访问记录播放一段时间后得到的文件系统场景作为文件系统状态,播放后续的访问记录作为测试负载.这种方法的优点是能够提供真实的访问请求,然而缺点也很突出:它生成的分级存储状态非常不真实,导致测试结果具有误导性.由于文件访问存在局部性,生成的文件只是真实文件系统中很小的一部分,尤其是必然会丢失那些较长时间不用的文件.这与分级存储系统的应用场景不相符.

本文给出了一个分级存储系统性能测试工具 DMStone,它使用文件系统快照生成某一时刻的系统状态,并根据后续的相邻快照的差异分析出后续访问负载的宏观特征,进而生成后续对文件系统的数据访问.DMStone 能够提供某一时刻完整的文件系统状态,涵盖了近期访问过的和长期不用的所有文件,而且它能够保证后续文件访问的局部性与真实应用场景相符合.

我们在 Linux 平台上实现了 DMStone,实验结果表明,DMStone 能够生成满足分级存储系统测试要求的文件系统真实场景,根据用户的配置灵活生成文件访问负载,能够对分级存储系统进行全面而合理的测试.应用 DMStone 对分级存储系统 AIP 进行了测试,验证了 DMStone 性能测试的有效性.

1 相关工作

分级存储系统性能测试需要向具有真实状态的存储系统播放有代表性的访问负载,两个主要技术挑战是创建真实的文件系统状态和生成有代表性的 I/O 负载.在 2008 年的综述文章[13]中,Traeger 等人查阅了 4 个国际顶尖会议 SOSP,OSDI,FAST,USENIX 从 1999 年~2007 年这 9 年时间内的全部论文,调研了其中存储相关的 106 篇论文中提到的 415 种文件系统和存储测试方法或工具.我们发现,这些方法都毫无例外地忽略了对存储系统状态的构建.

在文件系统场景构建方面,Agrawal 和 Arpaci-Dusseau 等人[14]提出一种构建方法.它根据一定的模型生成相应的目录树结构,并将具有不同大小、不同扩展名的文件按照一定方式在这些文件夹之间分布.在处理文件在目录树结构中的分布情况时,对文件内容及数据在磁盘上的分布进行处理,以提供用户指定的文件系统场景.该方法的缺点是没有考虑 I/O 负载和系统状态的匹配问题,同时,没有对文件的时间、用户等属性分布进行研究,不能生成适宜测试分级存储系统的场景.

在文件访问负载上,Iometer[15]通过配置读写比例、随机读与随机写的比例、突发请求次数、两次突发请求的时间间隔等参数生成访问负载,模拟存储器或网络的 I/O 负载情况,对存储器或网络控制器的带宽、延时吞吐量等进行评测.而 SynRGen[10]文件访问负载生成器在系统调用层实现,它通过访问获取应用的特性,对应用的负载情况进行模拟.Fstress[9]是一个支持 NFS v3 的负载生成器,通过设置文件和文件夹的分布情况、符号链接的个数、目录树的最大深度、新建文件的访问情况、文件大小、I/O 大小等生成相应负载.目前的负载生成器

在生成负载时,并没有考虑文件访问在具有不同时间属性的文件之间的分布规律,但在分级存储的应用系统中,大部分文件访问应该集中在最近创建的文件上.

2 DMStone 的设计

2.1 系统架构

DMStone 使用文件系统快照生成存储系统某一时刻的系统状态,并根据后续的相邻快照的差异分析出期间访问负载的宏观特征,进而生成后续对文件系统的访问请求.如图 1 所示,DMStone 由 4 部分组成,分别为快照读取模块 Retriever、系统状态重建模块 Reconstructor、访问请求增强模块 Reinforcer 以及访问请求播放模块 Replayer.

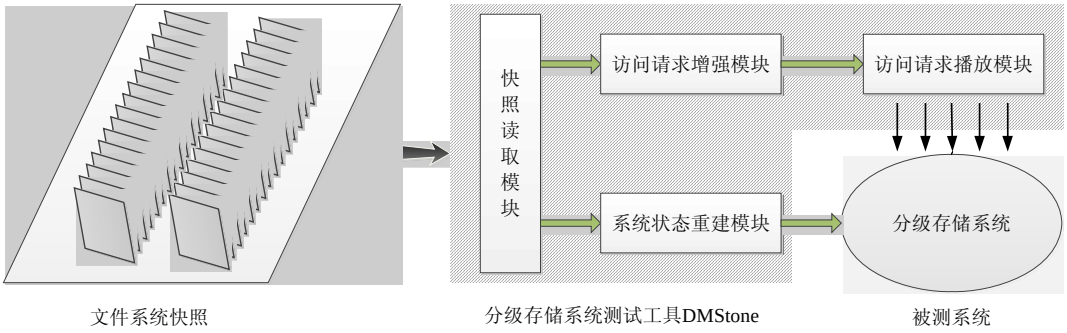


Fig.1 Architecture of DMStone

图 1 DMStone 组成结构

在对指定的分级存储系统进行测试的过程中,DMStone 的 4 个模块依次执行如下动作:

(1) 快照读取模块根据指定的时间点扫描快照数据,根据快照链表结构提取此时间点的状态描述信息并保存到指定文件中.同时,根据指定时间点快照及其后继快照信息生成文件访问请求特征信息;

(2) 系统状态重建模块依据生成的状态描述信息及用户设置的参数,在指定的文件系统中生成文件系统场景;

(3) 访问请求增强模块根据快照读取模块生成的访问请求特征信息,对文件系统的访问请求进行细化处理,增强访问负载的强度;

(4) 访问请求播放模块在生成的文件系统场景的基础上,播放增强后的文件访问负载,同时记录请求的响应时间等性能统计数据.

2.2 被测系统的状态重建

DMStone 根据某一时刻的文件系统快照信息生成文件系统场景,这些快照信息包括文件路径及文件名、文件大小、文件的最近访问时间、文件的最近修改时间、用户名等.将快照信息使用文件记录的形式来描述有利于生成后续的文件访问请求.由于文件的内容与系统评测并不相关,所以,某一时刻的快照可由文件系统中所有文件的相关记录信息来描述,从而便于被测系统的重复创建,进行多次测试.

在生成文件系统场景描述文件后,DMStone 根据这些信息生成文件系统场景,并保证系统中文件的属性与快照中的信息一致.对每条文件记录,选择文件的最近访问时间和最近修改时间中的较小值作为文件的创建时间.为了保证文件的最近访问时间和最近修改时间与快照中的相一致,在文件创建以后,根据文件的最近访问时间和最近修改时间中的较大值进行一次文件读写操作,并保证除文件的相应时间属性以外,其余基本属性不发生变化.如果最近访问时间值较大,在创建文件后,根据最近访问时间再对文件进行一次读操作.如果最近修改时间值较大,则依据这个最近修改时间,对文件进行一次写操作.对文件内容而言,可以随机写入数据.

2.3 负载信息的生成

生成基于已有文件系统场景的文件访问负载,是 DMStone 的另外一个重要部分.DMStone 生成的文件访问负载能够反映文件访问的局部性,并体现实际应用中的文件访问特征.生成文件访问负载包括两个步骤:生成基本访问负载以及对基本访问负载进行负载强度增强处理.

2.3.1 基本访问负载的生成

基本访问负载信息需要确定所访问的文件集合以及每个文件的访问类型.将每个文件访问请求用以下几个方面来描述:访问类型、文件路径名、访问起始地址、访问大小、访问时间及请求用户.DMStone 通过对比相邻两个快照之间的差异得到基本访问负载信息.由于生成快照会给原系统带来额外的 I/O 开销,所以如果生成快照的间隔太小,则会对系统的 I/O 性能产生影响;而如果快照生成时间间隔太大,则会影响获取的负载特征中创建文件及删除文件请求的访问执行时间的准确性.出于这样的考虑,DMStone 使用的快照时间间隔为 1 天.

一个文件在两个相邻快照之间是否被访问过,是通过对比相邻快照之间文件的元数据得到的.如果文件的属性有变化,则说明文件在这段期间有相应的访问请求.例如:如果文件的最近访问时间发生变化,则说明在这两个快照之间至少有一个对文件的读访问.文件的创建与删除请求,根据文件在前后快照之间是否存在来确定.

DMStone 根据访问类型和从快照中可以得到的信息量的不同,确定不同类型的文件访问的起始位置、偏移大小和执行时间等访问信息.例如对写请求,将前一快照的文件大小作为偏移起始地址,并将两个快照中文件大小的差值作为偏移大小;将后一快照中文件的最近修改时间作为文件访问的执行时间.而创建和删除请求的执行时间不能通过比较快照获得准确的时间点,只能在两个快照的时间段中随机生成一个时间点作为请求的执行时间.DMStone 通过将后一快照的最近访问时间和最近修改时间作为访问请求的执行时间,保证文件在时间属性上的误差在 1 天以内.分级存储系统处理的数据往往是最近访问在几十天以前或几百天以前的,所以 1 天以内的误差对分级存储系统评测的影响很小.

2.3.2 访问负载的增强

基本访问负载并没有包括所有应当被访问的文件,且对单个文件而言,每种访问类型最多只包含一个访问请求.例如,若在两个快照之间文件的最近访问时间有变化,则基本访问负载中仅有一个此文件的读请求.为了还原有代表性的访问负载,需要对这些访问负载进行强度增强处理.为了完成这个过程,需要确定被访问的文件、各种访问的数量、访问的分布情况、每个访问的起始地址及偏移大小.

首先,在生成访问请求的两个快照时间点之间创建、访问并删除的临时文件并没有包含在生成的基本负载信息中.针对临时文件的读写特性,Floyd[16]进行了详细的研究,并得到如下一些结论:(1) 临时文件的数量与实际读写文件的数量之间成一定比例;(2) 大部分临时文件的生命周期小于 1 分钟;(3) 临时文件的文件大小一般不到 1KB,且只有一次或两次的读写.依据这些结论,我们通过基本访问负载中的读写文件数量确定临时文件的数量;其文件创建时间在访问开始和结束时间点之间随机生成,并根据其生命周期性质决定文件的删除时间;在文件创建以后,等概率地对其进行一次或两次的文件读写操作.

其次,基本访问负载中,对每个文件的访问,每种访问类型最多只有一个,所以对每个被访问文件的每种访问,都需要根据快照提供的不同信息进行负载增强处理.其中,创建和删除文件访问请求与原始文件访问负载中的情况相同.在 Plan9[17]文件系统的快照中,文件的写次数可以通过相邻快照间文件的版本号的差值得到.为了适应不同的应用请求,DMStone 根据用户配置的读写访问比例确定文件的读写次数.

文件访问负载具有突发性,大多数访问集中在某一个时间段,而较小的部分分散分布.Wang 等人通过实验得到大多应用都具有 I/O 访问突发特性,且 65%~100%的写请求集中在一段时间区域[18].为体现文件访问的突发特性,DMStone 首先在文件创建时间或当日访问请求开始时间和读写此文件的访问执行时间之间确定一个时间点作为突发访问集中区域.在增强过程中,以这个时间点为中心,每隔 $1/P_i$ 秒添加一个相应的文件访问请求,直到文件的访问请求个数达到预期值或者时间点到达原始文件的访问执行时间点为止.在添加访问请求的过程中,P_i 值每经过一步,均按照如下方式来变化:

$$P_i = P_{i-1} \times D_i,$$

$$D_i = \begin{cases} D_o, & \text{if } P_i > P_{\max} \\ 1/D_0, & \text{if } P_i < P_{\min} \\ D_{i-1}, & \text{otherwise} \end{cases},$$

其中,$P_{\max}$ 为用户设置的最大 P_i 值,且 $P_0=P_{\max}$;而 $P_{\min}$ 为用户设置的最小 P_i 值,$1/P_i$ 为添加的第 i 个访问与前一个访问之间的执行时间间隔;D_i 代表 P_i 的变化系数;D_0 为用户设置的初始值,且 $D_0 \in (0,1)$,当 $D_0=1$ 时,表示请求将按照恒定的时间间隔访问文件.用户根据期望的文件访问时间间隔设置 $P_{\max}$,$P_{\min}$ 和 D_0 的值,从而得到与应用具有相同负载特征的访问请求.

文件的读写访问请求区分为随机读和顺序读、随机写和顺序写.DMStone 根据指定的参数确定随机读写与顺序读写之间的比例,并根据这个比例确定文件的读写请求的类型.对每个读写访问请求,需要确定读写访问请求的起始地址和偏移大小.随机读写访问请求的起始地址及偏移大小在文件大小范围内随机生成;顺序读写访问请求的起始地址从文件起始位置开始,偏移大小按照等概率从 512B~64KB 之间得到.

2.4 访问负载的播放

DMStone 在评测分级存储系统的过程中,首先在生成的文件系统场景上运行分级存储系统,分级存储系统将依据分级规则,将文件迁移到二级或三级等低级存储设备中,而将具有较高文件价值的文件保留在一级存储设备上.在对文件进行分级处理后,DMStone 播放增强后的访问负载来完成分级存储系统的性能评测.

在真实系统状态上播放后续的文件访问请求时,DMStone 根据指定参数决定负载播放的时间长短.文件访问线程每次读取一个文件访问请求,当请求播放的时间点到达时,执行对文件的访问,同时记录请求类型及请求响应时间.在评测结束时,DMStone 对不同类型的数据进行处理,生成相应的数据信息,完成系统性能数据统计.

3 DMStone 的特点与使用

将 DMStone 应用在分级存储系统 AIP 中,对 AIP 进行全面的评测.在评测过程中,DMStone 使用了贝尔实验室的 Plan 9 文件系统快照[17],它包含从 1990 年~2001 年间每天的系统快照信息.在实验中,DMStone 还原了 1993 年 5 月 31 日的文件系统场景,同时生成此后一天的文件访问请求.

3.1 文件分布

为了验证 DMStone 恢复系统状态的真实性,我们对还原得到的 1993 年 5 月 31 日的文件系统场景进行了统计分析.图 2 给出了具有不同最近访问时间、不同最近修改时间的文件数的累积分布.

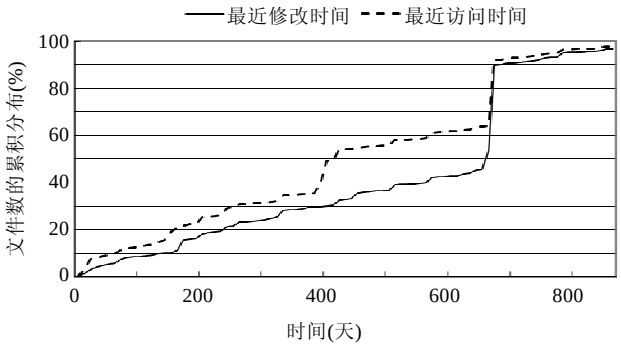


Fig.2 File distribution according to file last change time and last access time

图 2 文件按照最近修改时间及最近访问时间的分布

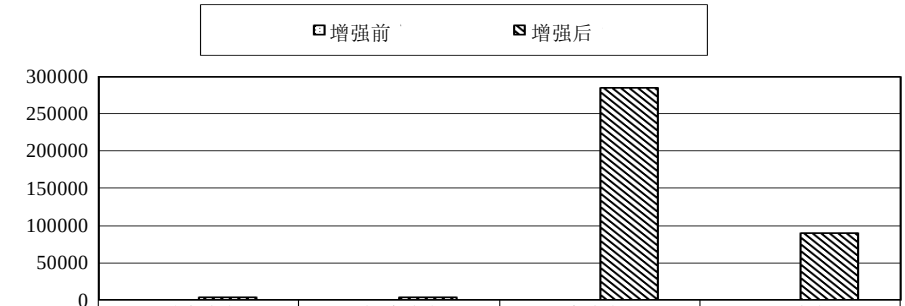
可以看到,最近 10 天内访问过的文件数只占总数的 2.8%,而最近 50 天内访问过的文件数占总数的 8.7%,100 天内被访问过的文件数占总数的 12.6%.最近 10 天内修改过的文件数占总数的 1.64%,50 天内修改过

的文件数占总数的 5.0%,最近 100 天内修改过的文件数占总数的 8.74%.这种情况与分级存储的目标应用相符.

3.2 访问请求比例

通过比较 Plan 9 文件系统在 1993 年 5 月 31 日与 1993 年 6 月 1 日的快照,可以得到被访问的文件,同时可以确定对每个文件的访问类型.根据需要,在对基本访问负载增强处理的过程中可以配置文件读写的比例.实验中,将读写比例设置为 4:1,P_{max} 设置为 10,P_{min} 设置为 0.1,递减速率 D_0 设置为 0.98,使得文件的读写分布在 500s 左右的时间范围内按照一定的形式分布.将临时文件与被读的文件之间的比例设置为 1:1.

图 3 对比了基本访问负载增强前后的情况.可以看出,增强处理后,删除请求与创建请求均增加了 3 449 个,这也是临时文件在系统中的个数.读请求在增强处理后与写请求的比例为 3.15:1.之所以比例与预设的 4:1 不同,是由于根据比例生成时,对某些文件会预期生成大量的读请求;但按照算法生成时,在超过文件的创建时间和文件最近访问时间时,增强处理过程会结束.



	删除请求	创建请求	读请求	写请求
增强前	330	472	3 449	646
增强后	3 779	3 921	284 924	90 499

Fig.3 File access requests before and after enhancement

图 3 访问请求增强前后对比

在生成的读写访问中,顺序读与顺序写的比例如图 4 所示.由于在增强过程中临时文件都是整个文件的读写,所以把临时文件的读写当成随机读写看待.在设置过程中,顺序读写与随机读写的比例设置为 4:1.增强处理后,随机读个数为 229 142 个,占读请求总数的 80.4%;随机写请求个数为 77 134 个,占写请求总数的 85.2%.造成这种差异的原因是,顺序读写是针对一个文件而言的,而读写请求在不同文件之间并不是平均分布,对其中一些文件的读写请求频繁,而对于另外的一些文件则读写较少.

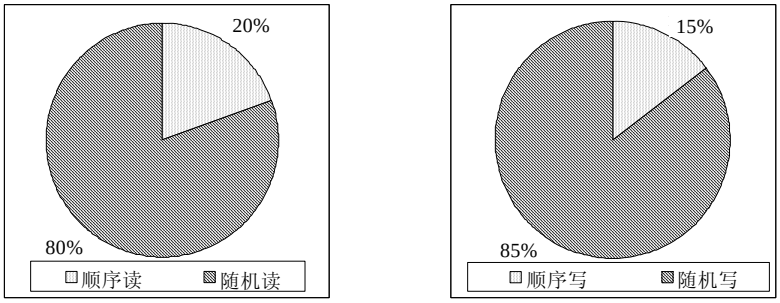


Fig.4 Ratios of sequential file access requests after enhancement

图 4 增强后顺序读写的比例

从结果可以看到,DMStone 能够根据配置动态生成符合用户需求的访问负载,体现了真实应用场景中的负载特性.

3.3 访问分布

为体现文件访问请求的突发特性,对于文件访问而言,按照一定的函数在一定时间区域内分布.实验中参数设置与上一实验相同,即 P_{max} 设置为 10,P_{min} 设置为 0.1,递减速率 D_0 设置为 0.98.这样设置使得每秒中访问量最多为 10 个,两个访问之间最大时间间隔为 10s.我们对经过增强处理的访问负载进行了分析,对其中一个访问请求数较多的文件进行了统计,每隔 10s 统计这个文件的读访问总数,得到这个文件的访问的分布按照规律周期性出现文件访问密集区域.其中,一个周期的分布情况如图 5 所示.

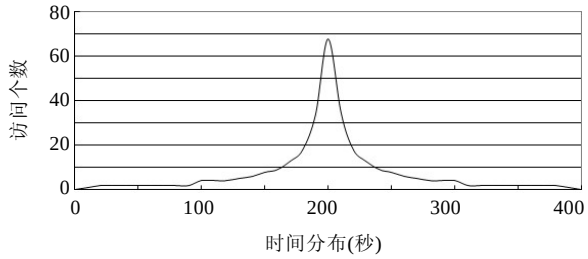


Fig.5 File access requests distribution during a fraction of time

图 5 文件访问在一段时间上的分布

图 5 中时间分布为相对时间.如同预期,对一个文件的访问能够按照指定参数体现访问请求的突发特性,在访问的集中区域,10s 之内对此文件的访问请求有 68 个;在大部分的时间区域内,访问呈分散分布.

3.4 分级存储系统实测结果

3.4.1 数据分级存储能力测试

用 DMStone 对分级存储系统 AIP 进行了性能测试.AIP 是一个基于策略的分级存储系统,它依据用户制定的数据管理策略,判断数据的价值,并将数据在不同存储层级之间进行迁移.将 AIP 运行在前面实验中生成的文件系统场景之上.

在实验中,使用三级存储池,其中假设主存储的容量为 9GB.使用的两条数据迁移策略内容如下:

- 策略 A:当主存储池的空间利用率大于 95%时,将最近访问时间是 10 天以前的文件迁移到二级存储池,直到主存储池的空间利用率小于 85%.迁移过程中,按照文件从大到小的顺序迁移文件;
- 策略 B:将最近访问时间是 150 天以前的文件,从二级存储池迁移到三级存储池中.

实验中,每隔 40s 记录 3 个层级存储池的空间使用情况.如图 6 所示,在分级存储系统 AIP 迁移策略的管理下,文件在不同存储池之间动态迁移.随着迁移策略的执行,主存储池空间使用率减少,而二级、三级存储池中的空间使用率增加,使得数据得以在不同存储池中分布.说明 DMStone 生成的文件系统场景能够体现分级存储系统的应用环境情况,并最终体现分级存储的效果.

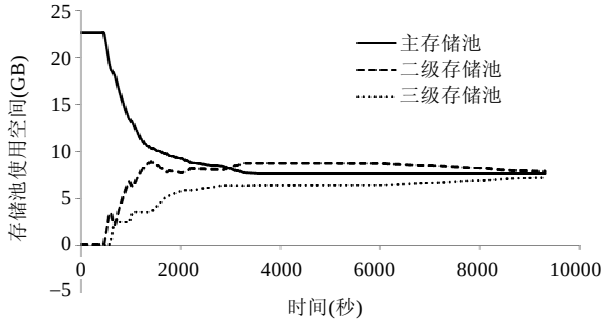


Fig.6 Space used of storage pools changed with time

图 6 存储池空间使用变化情况

3.4.2 I/O 响应能力评测

为了对 AIP 的文件 I/O 性能进行评测,我们通过 DMStone 分别对使用 AIP 和不使用 AIP 时文件系统的 I/O 性能进行统计.对使用 AIP 的情况,在运行 DMStone 之前,我们将文件场景中最近访问时间是 100 天以前的文件迁移到二级存储设备中,以模拟 AIP 应用后的场景.DMStone 在两种情况下均运行 30 分钟.在运行过程中,每隔 10s 统计 1 次 10s 内访问请求的平均响应时间,并在运行结束时统计其中读写比例、平均每秒的访问请求个数、1s 内最大请求数及所有请求的平均响应时间.其中,响应时间统计结果如图 7 所示.

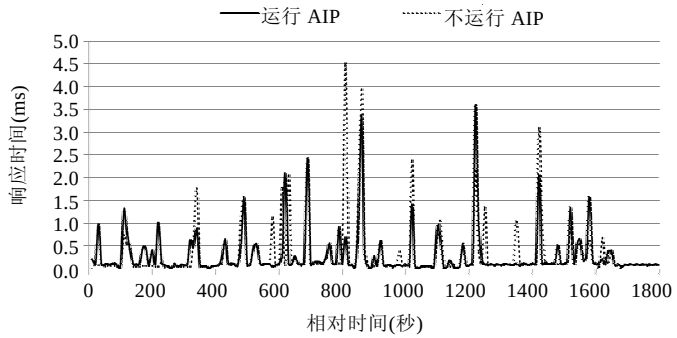


Fig.7 Comparion of response times with and without AIP

图 7 运行 AIP 与不运行 AIP 时 I/O 请求响应时间对比

将 DMStone 运行在不同情况下,可以评测分级存储系统是否会对原系统造成较大影响,从而确定分级存储系统是否适合相关应用场景.在运行 AIP 时,测试得到所有请求的平均响应时间为 0.375ms;而不运行 AIP 时,平均响应时间为 0.365ms.从实验结果中可以看到,应用 AIP 后,几乎不会对原系统的 I/O 性能造成影响,说明 AIP 适宜应用在 DMStone 所模拟的场景中.

4 结 论

已有分级存储系统评测方法通过播放一段时间的文件访问请求生成文件系统场景,因为彻底忽略了近期未被访问的文件而与分级存储的真实场景不符.本文提出了一个分级存储系统性能测试工具 DMStone,它使用文件系统快照生成某一时刻的系统状态,并根据后续相邻快照间的差异提取访问负载的特征,进而生成后续的 I/O 负载.

使用 Plan 9 文件系统快照的实验结果表明,DMStone 能够提供某一时刻完整的文件系统状态,涵盖近期访问过的和长期不用的所有文件.此外,DMStone 能够保证后续文件访问的局部性与真实应用场景相符合.我们应用 DMStone 对分级存储系统 AIP 进行了测试,验证了 DMStone 性能测试的有效性.

References:

[1] Gibson TJ, Miller EL, Long DDE. Long-Term file activity and inter-reference patterns. In: Proc. of the 24th Int'l Conf. on Technology Management and Performance Evaluation of Enterprise-Wide Information Systems. Anaheim, 1998. 976−987.

[2] Smith AJ. Long term file migration: Development and evaluation of algorithms. Communications of the ACM, 1981,24(8):521−532. [doi: 10.1145/358722.358737]

[3] Salmon E, Tarshish A, Palm N, Patel S, Saletta M, Vanderlan E, Rouch M, Burns L, Duffy DD, Caine R, Golay R, Paffel J, Schumann N. Hierarchical storage management at the NASA center for computational sciences: From unitree to SAM-QFS. In: Kobler B, Hariharan PC, eds. Proc. of the 12th NASA Goddard, 21st IEEE Conf. on Mass Storage Systems and Technologies (MSST 2004). Washington: IEEE, 2004. 177−183.

[4] Baker MG, Hartman JH, Kupfer MD, Shirriff KW, Ousterhout JK. Measurements of a distributed file system. In: Levy HM, ed. Proc. of the 13th ACM Symp. on Operating Systems Principles. New York: ACM, 1991. 198−212. [doi: 10.1145/121132.121164]

[5] Gribble SD, Manku GS, Roselli DS, Brewer EA, Gibson TJ, Miller EL. Self-Similarity in file systems. In: Leutenegger S, ed. Proc. of the 1998 ACM SIGMETRICS Joint Int'l Conf. on Measurement and Modeling of Computer Systems. New York: ACM, 1998. 141−150. [doi: 10.1145/277851.277894]

[6] Ousterhout JK, Costa HD, Harrison D, Kunze JA, Kupfer M, Thompson JG. A trace-driven analysis of the UNIX 4.2 BSD files system. In: Waite WM, ed. Proc. of the 10th ACM Symp. on Operating Systems Principles. New York: ACM, 1985. 15−24. [doi: 10.1145/323647.323631]

[7] Riedel E, Kallahalla M, Swaminathan R. A framework for evaluating storage system security. In: Proc. of the FAST 2002 Conf. on File and Storage Technologies. Berkeley: USENIX Association, 2002. 15−30.

[8] SNIA. Storage network industry association: Iotta repository. 2007. http://iotta.snia.org

[9] Anderson DC, Chase JS. Fstress: A flexible network file service benchmark. Technical Report, Duke University, 2002.

[10] Ebling MR, Satyanarayanan M. Synrgen: An extensible file reference generator. In: Gaither BD, ed. Proc. of the 1994 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems. New York: ACM, 1994. 108−117. [doi: 10.1145/183018.183030]

[11] Anderson E, Kallahalla M, Uysal M, Swaminathan R. Buttress: A toolkit for flexible and high fidelity I/O benchmarking. In: Proc. of the FAST 2004 Conf. on File and Storage Technologies. San Jose: USENIX Association, 2004. 45−58.

[12] Mesnier MP, Wachs M, Sambasivan RR, Lopez J, Hendricks J, Ganger GR, O'Hallaron D. Trace: Parallel trace replay with approximate causal events. In: Proc. of the 5th USENIX Conf. on File and Storage Technologies. Berkeley: USENIX Association, 2007. 153−167.

[13] Traeger A, Zadok E, Joukov N, Wright CP. A nine year study of file system and storage benchmarking. ACM Trans. on Storage (TOS), 2008,4(2):1−56. [doi: 10.1145/1367829.1367831]

[14] Agrawal N, Arpaci-Dusseau AC, Arpaci-Dusseau RH. Generating realistic impressions for file-system benchmarking. In: Seltzer MI, Wheeler R, eds. Proc. of the 7th Conf. on File and Storage Technologies. Berkeley: USENIX Association, 2009. 125−138. [doi: 10.1145/1629080.1629086]

[15] OSDL. Iometer project. 2004. http://www.iometer.org/

[16] Floyd R. Short-Term file reference patterns in a Unix environment. 2009. https://urresearch.rochester.edu/institutionalPublicationPublicView.action?institutionalItemId=6362

[17] Pike R, Presotto D, Thompson K, Trickey H. Plan 9 from Bell Labs. Computing Systems, 1995,8(3):221−254.

[18] Wang F, Xin Q, Hong B, Brandt SA, Miller EL, Long DDE, Mclarty TT. File system workload analysis for large scale scientific computing applications. In: Kobler B, Hariharan PC, eds. Proc. of the 21st IEEE Conf. on Mass Storage Systems and Technologies/12th NASA Goddard Conf. on Mass Storage Systems and Technologie. Washingtong: IEEE, 2004. 139−152.

丘建平(1987－),男,福建龙岩人,硕士生,主要研究领域为网络存储.

舒继武(1968－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络/云存储,存储安全与可靠性,并行/分布式处理.

张广艳(1976－),男,博士,讲师,主要研究领域为海量存储,分布式处理.