频率自适应的动态副本管理机制*)

周 旭 卢显良 侯孟书 詹 川
(电子科技大学计算机学院 成都610054)

摘 要 本文提出了一种新颖的基于访问频率的分布式文件系统自适应动态副本管理机制:FSRM(Frequency Sensitive Replica Management)。在FSRM中,节点周期性地扫描本地副本读写情况,根据系统对文件读写模式的变化,自主决定本地副本的增删或迁移,以调整文件副本数量以及副本存放位置,提供更高的系统性能,减少网络流量。同时,FSRM中各副本对应的扫描周期长度不固定,能根据本地副本被访问的频率自动增减,从而以更少的系统开销提供灵敏的系统适应性,同时实现对系统中各个副本实现不同精度的差异化管理,合理分布系统荷载。最后,文章通过实验结果分析了FRSM的实际性能,验证了FSRM管理机制的有效性。

关键词 分布式,副本管理,动态,频率,自适应

A Frequency Sensitive Dynamic Replication Management Mechanism

ZHOU Xu LU Xian-Liang HOU Meng-Shu ZHAN Chuan
(Department of Computer Science of UEST of China,Chengdu 610054)

Abstract This paper presents a novel distributed adaptive Frequency Sensitive Replica Management mechanism (FSRM). In FSRM,in order to improve system performance and reduce network traffic,every replica of a certain file checks its state periodically to find out the access mode during the period of time,then decides whether move,delete itself or copy a new replica to another node. The interval of periodicity of each replica is also variable,it can be changed according to the access frequency of a replica,so that we can improve performance greatly with less system overhead, and realize a multi-leveled sensitive dynamic replica management. In the end,the paper evaluates the efficiency of FSRM by experimental result and shows the performance enhancement.

Keywords Distributed,Replica management,Dynamic,Frequency,Adaptive

1 引言

文件副本技术是一种最为常用的分布式数据管理机制。通过为系统中的文件增加各种不同形式的副本,保存冗余的文件数据,可以十分有效地提高文件的可用性和可靠性,避免在地理上广泛分布的系统节点由于网络断开机器故障等动态不可测因素引起的数据丢失和不可得。同时,通过合理地选择系统节点放置副本,与适当的路由协议配合,可以实现文件数据的就近访问,减少访问延迟,提高系统性能。

但是,由于分布式文件系统中文件数据的一致性要求,必然会涉及在各个文件副本之间进行数据的同步操作。而随着系统规模的不断扩大,网络环境的复杂化,以及文件副本数量的不断增加,同步文件数据带来的网络开销和操作延迟也急剧增加,甚至可能抵消掉文件副本带来的性能提升。

所以在分布式文件系统中,副本管理机制必须能够解决两个关键的问题:副本数量和副本位置。副本数量(replication degree)太少,文件的可用性和可靠性得不到有效的保证,且大量的文件操作集中在少量系统节点,导致节点负荷过重,系统性能受到影响。而副本数量过多,不但将占用大量的系统存储资源,并且会大大增加网络中写操作带来的文件数据同步操作数量,占用系统带宽,增加操作延迟,增加副本管理的复杂度。副本位置(allocation of the replicas)决定了在某一时刻一个文件的多个副本应该分别存放在系统哪些节点上。副本分布位置合理可以使得节点发起的操作请求在本地完成,或者从靠近自己的节点处得到满足,这将大大减少系统的通信流量,加快系统对文件请求的响应速度。反之,如果副本位置分布不合理,一个文件请求往往需要流经多个网络节点之后才能得到满足,这不仅增加了对文件访问的响应时间,同时也造成了对网络资源的浪费。

针对以上这两个关键问题,近年来提出了多种不同的副本管理机制,总的来说可以分为静态副本管理策略和动态副本管理策略两类:

·静态副本管理策略[1](static replication policy)在文件创建的时候就决定了副本的数量和放置位置,且不会随着系统状态的变化而改变。这要求在文件创建之初,就能预先预测文件的访问模式和可能对这个文件进行操作的节点范围,从而可以将适当数量的副本分布在适当的系统节点中。文件一旦创建完毕,副本数量和位置就不再改变。

·动态副本管理策略(dynamic replication policy)则是在文件创建之后,在文件的整个生命周期之中,根据系统当前状态的变化不断地改变副本方案(replication scheme),包括副本数量的增减和副本位置的迁移,以尽量少的存储代价和网络开销,实现更好的系统性能。根据管理策略的具体决策方式,动态副本管理策略又可以进一步分为集中式和分布式两种类型。集中式(centralized)的副本管理需要对每个文件副本的存取情况有一个全局的认识(global view),据此来对当前的副本方案做出调整,统一制定新的方案,适应系统的变化[2]。而分布式(distributed)的副本管理则与之不同,每个副本根据自己被访问的情况自主决定在什么时候什么位置为该文件增加或减少副本,而不需要了解其他节点上副本的访问情况。

本文提出了一种新颖的基于访问频率的分布式文件系统

*)本文受国家95重点攻关项目支持。周 旭 博士研究生,主要研究方向:计算机网络、网络存储、分布式操作系统等。卢显良 教授,博士生导师,主要研究方向:计算机网络、操作系统。侯孟书 博士研究生,主要研究方向:计算机网络、网络存储。詹 川 博士研究生。

自适应动态副本管理机制——FSRM(Frequency Sensitive Replica Management),FSRM 不但能够根据文件读写访问频率,自动调整文件副本数量以及副本存放位置,平衡读写操作对副本的不同要求,并且还可以在文件的不同副本之间,根据副本各自被访问的频率,对每个副本实现不同精度的差异化管理。通过应用 FSRM 机制,我们可以更加灵活高效地管理文件副本,在不增加系统负载的条件下更灵敏地实现对副本的动态管理。

本文第2节首先介绍了其他研究者在文件副本管理机制方面做出的一些相关工作,第3节介绍 FSRM 的原理模型,第4节是 FSRM 的实现细节,在第5节我们通过实验结果验证了 FSRM 的有效性,最后是全文的结论。

2 相关工作

在上面提到的几种副本管理机制中,分布式动态副本管理策略是近年来受到关注较多的一个研究方向。通过在节点上为文件访问次数设定一个计数器,Bartal 等人实现了一种动态副本管理算法[3]:计数器随节点对文件的读而增加,反之随着对文件的写而减少,当计数器的值超过某个固定的阈值时,节点主动将文件在本地保留下来,成为一个新的副本,当计数器为0时,节点删除本地副本。这种管理机制实现简单,但是选取一个合适的阈值比较困难,并且所有访问文件的节点不论是否拥有副本,都必须为文件维持一个计数器,这无疑扩大了副本管理的节点范围,增加了系统开销。Giacomo 等人提出的另一种动态副本管理策略[4]则是在节点上以固定的周期运行一个扫描进程,当进程发现一个副本的读写比例超过一定界限之后,启动动态管理算法,决定增加或者减少一个副本。这种方式只涉及拥有副本的节点,节点管理范围小,实现也较为简单。但是实验表明,该算法对扫描周期的长度十分敏感,不同的扫描周期长度对系统的性能影响较大:在文件访问不频繁的情况下,如果保持较高的扫描频率,系统开销大,而当文件访问频繁时,如果扫描周期过长,算法将无法及时地反映系统的变化,导致性能下降。而在 Swarup 等人的模型中[5],节点收到的文件读写请求被划分为序列(sequence),并利用有限状态自动机(FSA)的自学习功能来预测该副本将要收到的下一个访问序列,从而提前作出相应的副本增删和迁移。这种方式对系统变化的适应性较好,能有效地减少系统中的通信流量,但是建立 FSA 的算法相对较为复杂,需要保存较大数量的其他节点的访问信息,如果系统规模较大,计算复杂度将急剧增加。

3 FSRM 模型描述

在研究了现有的动态副本管理机制的基础上,本文提出了一种新颖的分布式动态副本管理机制 FSRM。

为了建立 FSRM 的算法模型,我们假设系统对文件的读写操作按照 read one write all 的模式进行,即从一个副本读取而写回所有副本,以保持所有副本的一致性;假设系统的路由算法满足"就近读取"的原则,即可以使得一个没有副本的节点的读写请求从离它最近的副本处得到满足;我们还假设,一个节点在相邻周期的读写模式差别不大,即读写模式有一定的延续性,在一段时间内相对固定[6]。

3.1 副本增删条件

副本数量的增加将带来文件读操作的性能提高和文件写操作网络开销的增加,这是一对矛盾。一般来说,应该为读操作比较频繁的文件建立较多数量的副本,而为写操作较频繁的文件保持较少数量的副本,通过不断调整副本数量和位置,平衡读写操作的性能差异,实现更好的系统性能。

设系统中一个文件 f 有 n 个副本,以 node$_i$,$i=1\cdots n$ 表示存放这 n 个副本的系统节点。设 α 表示一个没有副本的节点从离它最近的副本读取文件数据的系统开销,β 表示写操作更新一个非本地副本的系统开销。由于我们的副本策略是分布式的,因此我们希望每个副本节点仅仅通过自身掌握的文件访问情况来自主决定是否应该增删副本。对一个时间周期 t,我们为每个副本节点 node$_i$,维持几个计数器,记录该周期内节点的副本读写请求:r_{in} 为 node$_i$ 内部发出的对文件读请求,$r_{out,j}$ 为 node$_i$ 收到的从另外一个节点 node$_j$ 发出的文件读请求,w_{in} 为 node$_i$ 内部发出的文件写请求,w_{out} 为 node$_i$ 收到的非本节点发出的文件更新请求。

因为我们假设一个节点对文件的访问模式在一段时间内变化不大,那么我们可以认为下一周期将要到来的文件访问请求与现周期的文件访问模式相同。由此可知,若 node$_i$ 决定在下一周期为文件在 node$_i$ 上增加一个副本,判断的条件是增加副本后的系统开销小于增加前的系统开销,即

$$r_{out,j}\alpha+nw_{out}\beta+(n-1)w_{in}\beta>(n+1)w_{out}\beta+nw_{in}\beta$$

化简后得

$$r_{out,j}\alpha>(w_{out}+w_{in})\beta \quad (1)$$

同理,如果 node$_i$ 决定在本周期结束之时删除自己的副本,判断的条件是删除副本后的系统开销小于删除前的系统开销,即

$$nw_{out}\beta+(n-1)w_{in}\beta>r_{in}\alpha+(n-1)w_{out}\beta+(n-1)w_{in}\beta$$

$$r_{in}\alpha<w_{out}\beta \quad (2)$$

在实际系统中,为了简单起见,我们可以认为 $\alpha\approx\beta$,则式(1)、(2)可以简化成

$$r_{out,j}>w_{out}+w_{in} \quad (3)$$

$$r_{in}<w_{out} \quad (4)$$

3.2 扫描周期变化规律

是否增删一个文件的副本,是拥有副本的节点在该副本的一个周期结束之后,扫描节点内部该文件副本对应的计数器值,根据式(3)、(4)计算决定的。根据前文阐述,采用不同的周期长度,对系统性能的影响很大。

为了解决这个问题,FSRM 引入了一种对文件访问频率敏感的可变周期算法。该算法的思想是当节点上某一个文件副本被访问的频率高的时候,节点对该副本的扫描周期自动缩短,从而可以更加及时地反映文件访问模式的变化,以对节点的访问模式做出相应反应;当节点上某一个文件副本被访问的频率低的时候,扫描周期自动延长,减少节点启动扫描进程的频度,从而减轻节点负载。这样,对于同一个文件的不同副本,由于被访问的频率可能不一样,各自的扫描周期也将不同,某些访问频率较高的副本周期短,某些访问频率低的副本周期长,从而实现了对文件副本的差异化管理。

设一个副本的周期形成一个周期序列 $t_1,\cdots,t_n$,记该副本在周期 t_i 内收到的所有访问请求数量为 A_i。设该节点副本当前周期为 t_n,则下一周期长度 t_{n+1} 的计算公式为:

$$t_{n+1}=t_n\cdot\frac{A_{n-1}/t_{n-1}}{A_n/t_n} \quad (5)$$

4 FSRM 的实现

式(3)、(4)、(5)给出了动态副本管理的数学模型基础,下面将讨论 FSRM 的具体实现细节。

设系统中一个文件 f 的当前副本数量为 *replica_degree*,这个变量在文件的生命周期内一直处于不断变动之中。但是,*replica_degree* 的变动的范围应该加以限制,若 *replica_de-*

gree 太小甚至减少至0，则无法保证文件的可用性，若 *replica_degree* 无限制地扩大，又将会占用过多的系统存储空间。我们可以根据文件的可用性要求为它设置一个下限 *MIN*，根据系统资源状况为它设置一个上限 *MAX*[7]。

$$MIN \leqslant replica\_degree \leqslant MAX \quad (6)$$

每个拥有文件 f 副本的系统节点 i 维持3个计数器 r_{in}、w_{in} 和 w_{out}，分别记录当前周期内该副本收到的来自节点内部的读、写次数和来自节点外的写次数。同时节点 i 维持一个链表 outside_read_list，链表的每个单元记录了在本周期内，一个到节点 i 读取文件 f 的节点的信息和读取的次数，链表单元结构如图1所示，表示节点 j 在本周期内访问了节点 i 上的文件 f 副本共 $r_{out,j}$ 次。链表单元是按照访问副本节点的先后顺序排列的，处在链表头部的单元始终是最近一次访问文件副本的节点，即每当有一个外来节点 j 读取副本的时候，先在链表中搜索在本周期内是否已经读取过该副本，如果已读过，则将计数器 $r_{out,j}$ 的次数加1，并将该链表单元移至链表头部；若在本周期内该节点还没有读过副本，则在链表头部新插入一个单元，将读取次数置为1，nodeID 置为自己的 ID。

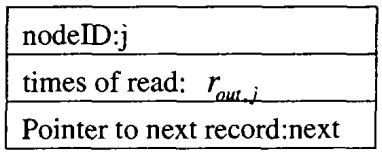


图1 链表结构图

为了实现对周期的动态调整，节点 i 维持一份对文件副本当前周期和前一个周期相关情况的记录，记录中保存了当前周期的长度 t_n、前一个周期的长度 t_{n-1} 和前一个周期内该副本收到的所有访问次数 A_{n-1}。其中副本收到的所有访问次数为 r_{in}、w_{in}、w_{out} 与所有的 outside_read_list 中 $r_{out,j}$ 之和。这样，就可以使用式(3)来计算节点的下一周期长度。

关于周期长度，还有几种特殊情况需要考虑。首先如果该副本在本周期没有访问，即式(5)中的 $A_n=0$，则不管上周期的读些次数为多少，下一周期长度增加为现周期长度的2倍；如果副本在上周期内没有访问，则不管本周期内访问次数为多少，下一周期长度减少到现有周期的一半。为了避免周期过短或过长，还应该为它设定一个范围，上限和下限分别记为 *MIN_PERIOD* 和 *MAX_PERIOD*，所有周期长度 t 都应该在这个范围之内，即

$$MIN\_PERIOD \leqslant t \leqslant MAX\_PERIOD \quad (7)$$

当节点 i 上文件 f 的副本的一个周期到期之后，节点将判断是否为该文件增删或者迁移副本。

节点 i 首先判断是否需要为一个新的节点 j 增加一个副本。节点从头扫描 outside_read_list，找到第一个满足式(5)条件的外来读节点，为它增加一个新的副本。由于最近一次读取过副本的节点会继续读取的可能性很大，因此这样的选择方法就保证了文件往满足条件且最有可能再次读取的节点复制。

如果副本没有被增加，则紧接着进行是否删除节点 i 本地副本的判断。当该副本的计数器满足式(4)的时候，并不能马上删除自己的副本，因为如果系统中同时有多个节点上的副本满足删除条件的话，不经协商而自行删除有可能导致副本数小于 *MIN*，甚至副本全部被删除。为了避免这种情况的发生，我们让副本在满足式(4)条件时，不马上删除，而是向其他有副本的节点发出删除请求，然后自身进入 deleting 状态。当其他节点收到删除请求时，如果自身处在 deleting 状态，则不回答该请求，反之则回复同意删除的消息。如果节点 i 收到的同意删除消息的数目大于 *MIN*，则立即删除自身的副本，如果节点 i 到了下一周期结束时，仍然没有收到足够的同意消息，则放弃删除，结束 deleting 状态，返回正常状态，即 deleting 状态最长只能持续一个周期。处于 deleting 状态的副本仍然可以接受来自其他节点的访问。

一个副本总是先后经历是否增加和是否删除两次判断，这种方式使得一个副本有可能在第一次判断之中先为别的节点增加一个副本，而在第二次判断中接着又删除自己的副本。这可以看作是一种变相的副本迁移行为，使得副本的管理更为灵活。

如果增删副本都没有成功且文件副本的数目已经达到最大值 MAX 时，FSRM 将采取真正的副本迁移的方式：扫描列表，找到第一个外来读次数大于节点 i 本地读次数的节点 j（即满足式(8)），在该节点上建立一个副本，并将节点 i 上的副本删除。这样做保证了在副本数不能继续增加的时候，副本方案仍能根据系统对文件的访问方式调整副本分布位置。

$$r_{out,j} > r_{in} \quad (8)$$

还有一种特殊情况，因为文件的访问可能存在一定的时效性，一个文件在经过一段时间的访问之后，可能以后就很少被访问甚至根本不被访问了。特别是经过一段时间的密集访问之后，文件的副本数可能已经增长到一定的数量，如果以后不再访问，那么为其保留太多的副本已经没有必要。而在这种情况下，按照式(3)、(4)、(6)的条件，副本数将保持不变，故我们还需要引入一个新的判断机制来解决这个问题。因为若副本长期不被访问，周期会逐渐延长至 MAX_PERIOD，如果一个节点上的副本在经历了连续的2个长度为 MAX_PERIOD 的周期之后，仍然没有任何访问，FSRM 会将该副本转入 deleting 状态，试图删除该副本。这个判断应该在增加和删除副本之前进行，也就是说满足长时间没有访问条件的副本就没有必要再进行副本增删迁移的判断了，直接进入 deleting 状态。

最后，在所有判断完毕之后，节点根据式(5)设置副本的下一周期，并为副本的各计数器清0。

5 实验结果

5.1 实验环境及参数

我们在校园网环境中，选取了分布在不同位置的10台机器，构建了一个 FSRM 的实验环境，网络拓扑图如图2所示。

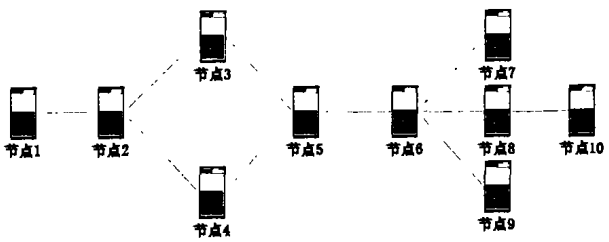


图2 实验系统拓扑图

为了测试 FSRM 的性能，我们对一个特定的文件 f 同时施行三种不同的副本管理测策略，进行对比试验。一种是标准的 FSRM 动态副本管理策略，一种是将 FSRM 中周期固定的策略（称为半动态副本管理策略 non-FRSM），最后一种是静态副本管理策略（即副本数量和位置都不变化）。其中 FSRM 的相关参数确定如下：MIN＝2，MAX＝8，MIN_PERIOD＝

20秒，MAX_PERIOD＝30000秒。静态副本管理策略中的副本数我们固定为3。

为了模拟在复杂网络情况下的系统行为，我们将系统中的节点分为几类：一类节点的文件访问模式按照一定的规律周期性地变化，一类节点访问模式固定不变，其余的节点完全随机地生成文件访问请求。所有节点的访问请求均按照"就近读取"的原则在本地或最近的副本处进行操作。

5.2 实验结果

我们设 non-FSRM 的扫描周期与 FSRM 的最小周期长度一样，都是20秒。运行试验系统一段时间后，统计在这段时间内系统中所有在网络中传递的操作总数，即节点到异地读取文件 f 和文件 f 同步更新的操作数之和。结果如表1所示。

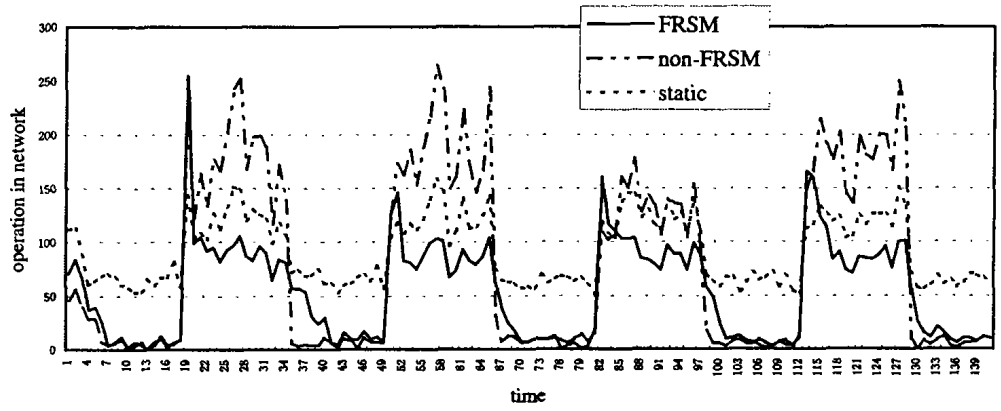
表1

	FSRM	Non-FSRM	Static
操作总数	26827	39472	44663

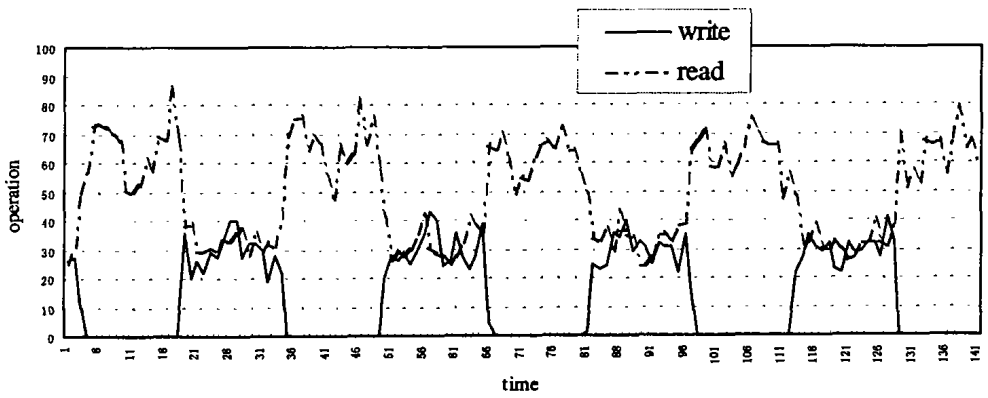
从结果中我们可以看出，FSRM 的通讯流量大约只有静态副本管理机制的60%，相对半动态副本管理策略，也只是它的67%，性能提高明显。而半动态副本管理策略却只比静态策略减少了5191次操作，性能有所提高，但不是十分明显。

为了更加深入地分析两种动态管理策略性能差异的原因所在，我们截取了一段时间内系统中操作数和副本数量变化的曲线图，如图3所示。其中图3.a 中的三条曲线分别记录了三种副本管理策略导致的在网络中传递的操作数，图3.b 记录了对应时段内的系统发出的对文件 f 的读/写操作数，图3.c 记录了对应时段内 FRSM 和 non-FRSM 各自的副本数量变化。

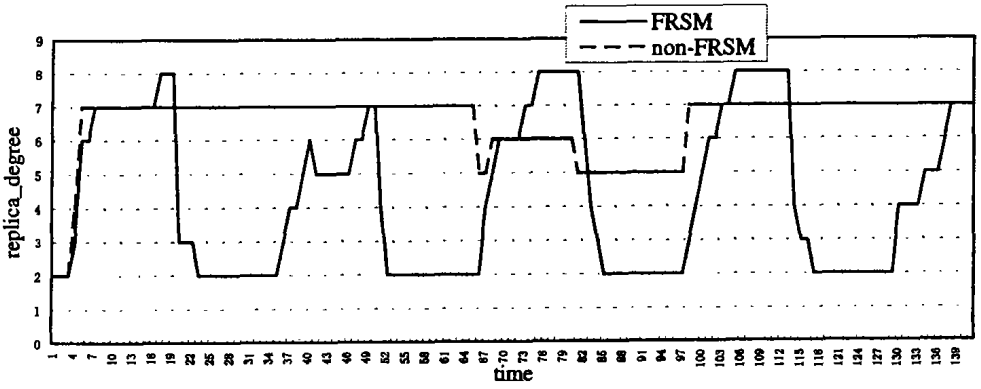
从图3.a 中我们可以发现，FSRM 和 non-FSRM 的网络操作数曲线都存在低谷（如5～18区间段），且两者的低谷位置是差不多完全重叠的。对应到图3.b 中，我们知道这些时间段正是系统对文件 f 发起大规模读操作的时段，而在图3.c 中，在这些时段中，两种动态管理策略对应的副本数都比较多。这说明 FSRM 和 non-FSRM 都察觉到了系统对文件读操作的增加，并相应增加了自己的副本数使得读操作尽量在本地得到满足，消化了在网络中传递的操作数。



a. operation in network



b. operation on file



c. replica_degree

图3 实验结果对比图

（下转第165页）

up property. In: Proc. of the IEEE Symposium on Research in Security and Privacy, 1987
10 McCullough D. Noninterference and the composition of security properties. In: Proc. of the IEEE Symposium on Research in Security and Privacy, 1988
11 McLean J. Security models and information flow. In: Proc. of 1990 IEEE Symposium on Research in Security and Privacy, IEEE Press, 1990. 177～186
12 Milner R. Communication and Concurrency. Prentice-Hall, 1989
13 Milner R. Communicating and Mobile Systems: the Pi-Calculus. Cambridge University Press, 1999
14 O'Halloran C. A calculus of information flow. In: Proc. of First European Symposium on Research in Computer Security (SORICS 1990), 1990. 147～159
15 Roscoe A W. CSP and determinism in security modeling. In: Proc. of the 1995 IEEE Symposium on Security and Privacy, IEEE Computer Society, 1995. 114～127
16 Roscoe A W. The Theory and Practice of Concurrency. Prentice-Hall, 1997
17 Roscoe A W, Woodcock J C P, Wulf L. Non-interference through Determinism. In: Proc. of European Symposium on Research in Computer Security 1994 (ESORICS'94), LNCS, Vol. 875, Springer-Verlag 1994. 33～53
18 Ryan P Y A. A CSP formulation of non-interference and unwinding. Presented at CSFW 1990 and published in Cipher, Winter 1990/1991. 19～30
19 Ryan P Y A. Mathematical models of computer security. In: Foundations of Security Analysis and Design - Tutorial Lectures (R. Focardi and R. Gorrieri Eds), LNCS, Vol. 2171, Springer-Verlag, 2001. 1～62
20 Ryan P Y A, Schneider S A. Process algebra and non-interference. Journal of Computer Security, 2001, 9(1,2): 75～103
21 Schneider S A. May Testing, Non-interference and Compositionality: [Technical Report CSD-TR-00-02 2001]. Royal Holloway, University of London
22 Stirling C. Modal and temporal logics for processes. LNCS, Vol. 1043, Springer-Verlag, 1996. 149～237
23 Sutherland D. A model of information. In: Proc. of the ninth National Computer Security Conf. 1986. 175～183
24 Wittbold J T, Johnson D M. Information flow in nondeterministic systems. In: Proc. of the 1990 IEEE Symposium on Research on Security and Privacy, 1990. 144～161
25 Zakinthinos A, Lee S. A general theory of security properties. In: Proc. of the 1997 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, 1997. 94～102
26 周伟, 尹青, 王清贤. 计算机安全中的经典模型. 计算机科学, 2004, 31(3)

(上接第136页)

图3.a 中 FSRM 曲线同时也存在数个突起(如19～37区间段),对应图3.b 中,我们可以知道这些正是系统对文件写操作比较频繁的时段。从图3.c 中可以看到 FSRM 自动减少了副本数量,避免大量的文件同步更新操作占用网络资源,且图3.b 中的系统写操作高峰期和图3c 中副本数量低谷吻合得比较好,证明了 FSRM 正确地反映了系统对文件读写模式的变化。仔细观察,在图3.a 中 FSRM 的每个突起的开始处,都有一个尖峰出现(如时间点19处),这是由于系统写操作的突然增加,FSRM 还来不及反应所造成的。但是我们也看到,FSRM 及时地调整了副本数量,使得网络中的操作数迅速减少,从而形成了这样的一个尖峰。这恰恰表明了 FSRM 对系统行为反映的灵敏度。

观察图3.a 中 non-FSRM 的操作数曲线,同样可以发现存在数个突起,但是这些突起没有迅速下降到一个合理的水平,而是保持较高的高度。对应 c 中我们可以知道这是由于 non-FSRM 的副本数量没有减少造成的。分析其原因,主要是因为各个副本的周期长度都相同,基本上都同时到期。于是各副本都发现系统写操作的增加,几乎同时提出删除自身的请求。由于同时提出请求的副本过多,反而不能收集到足够的同意应答,导致删除自身副本的操作失败,副本数维持在一个较高的水平。而反观 FSRM,由于各副本的周期各不相同,减少了同时提出删除请求的可能,因此使得副本数量变化更加灵敏。

再来看副本数量的平均值:FSRM 在实验时段内平均保有4.26个副本,而 non-FSRM 有6.22个,明显高于 FSRM。特别地,FSRM 的平均副本数4.26大于静态管理策略的3个,网络操作数反而远小于静态,也表明了 FSRM 能十分有效地提高系统性能,减少网络流量。

通过记录节点5上文件 f 副本的扫描周期,可以知道,在实验的时段内,节点5扫描文件 f 的本地副本共57次,最长周期达到了2233秒,最短周期为20秒,平均周期长度为176秒。而在 non-FSRM 中,节点5共扫描了500次,远远高于 FRSM 的次数。这说明了 FSRM 能以更小的系统代价获得更好的系统性能。

若将 non-FRSM 的扫描周期定为40秒,可以预测,由于扫描周期的放大,non-FSRM 更加不能迅速地反映系统操作行为的变化,性能将比20秒时更低。表2是经过与上节相同长度的时间段试验之后的结果,从表中我们可以看出 FRSM 保持了良好的性能,而 non-FSRM 性能下降,几乎与静态管理策略一样了。

表2

	FSRM	Non-FSRM	Static
操作总数	29061	41687	41273

结论 本文在分析现有的分布式文件系统动态副本管理机制的基础上,提出了一种新颖的基于访问频率的分布式文件系统自适应动态副本管理机制——FSRM。在 FSRM 中,节点周期性地扫描本地副本的读写情况,根据系统对文件读写模式的变化,自主决定本地副本的增删或迁移,以调整文件副本数量以及副本存放位置,提供更高的系统性能,减少网络流量。FSRM 的副本扫描周期长度不固定,能根据本地副本被访问的频率自动增减,以更少的系统开销提供灵敏的系统适应性,并可以对系统中各个副本实现不同精度的差异化管理,合理分布系统荷载。

通过实验,我们证明了 FSRM 能在不增加系统负载的条件下,更加灵活高效地实现对文件副本的分布式动态管理,提高系统性能,减少网络开销,是一种有效的动态副本管理机制。

参 考 文 献

1 Dowdy D, Foster D. Comparetive models of The File Assignment Problem. Computing Surveys, 1982, 14(2)
2 Bartal Y, Fiat A, Rabani Y. Competitive algorithms for distributed data management. In: Proc. 24th ACM Symp. on Theory of Computing, 1992. 39～50
3 Cabri G, Corradi A, Zambonelli F. Experience of Adaptive Replication in Distributed File Sytems. In: Proc. of EUROMICRO-22, 1996. 459～466
4 Acharya S, Zdonik S B. An Efficient Scheme for Dynamic Data Replication. Brown University CS-93-43, 1993
5 Wolfson O, Jajodia S, Huang Y. An adaptive data replication algorithm. ACM Transactions on Database Systems, 1997, 22: 255～314
6 Ranganathan K, Iamnitchi A, Foster I. Improving Data Availability through Dynamic Model-Driven Replication in Large Peer-to-Peer Communities. In: Proc. of the Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems, Berlin, May 2002