双粒度协议下基于hint的启发式缓存替换算法*

王建勇[1,3] 祝明发[2,3] 徐志伟[2,3]

[1](北京大学计算机科学与技术系 北京 100871)
[2](国家智能计算机研究与开发中心 北京 100080)
[3](中国科学院计算技术研究所 北京 100080)

摘要 合作式缓存技术是提高机群文件系统性能的关键技术之一. s2fs(scalable single-image file system)是一个单一映像机群文件系统原型,它利用双粒度协议实现了符合严格UNIX语义的合作式缓存. 该文为s2fs设计了基于hint的启发式缓存替换算法,并为其建立了性能分析模型. 分析结果表明,同现有的合作式缓存替换算法N-chance相比,启发式算法几乎在所有情况下都有效地降低了I/O的响应时间.

关键词 启发式缓存替换算法,N-chance算法,双粒度协议,s2fs(scalable single-image file system),合作式缓存.

中图法分类号 TP316

随着越来越多的分布式系统利用高性能网络将各计算节点连接起来,远程内存正日益成为一种新型的内存结构层次,因为通过高带宽、低时延的网络对远程内存的访问要比本地磁盘快得多. 这种远程缓存结构(remote caching architecture)[1]强调了各存储层次(即本地内存、远程内存和磁盘)之间性能上的差异. 由于它允许系统中所有节点互相存取各自的本地内存以充分利用远程内存,这种对称式结构显然不同于传统的客户/服务器模型,被许多文献[2,3]称为合作式缓存(cooperative caching).

国家智能中心正在研制的机群文件系统——COSMOS. 为了保证在提供单一系统映像功能的同时,达到提高I/O可扩展性的目的,也采用了合作式缓存等技术. 目前,COSMOS系统的原型已开发完毕,为了与未来成熟的COSMOS系统相区别,我们称目前的COSMOS原型系统为s2fs(Scalable Single-image File System).

1 s2fs及其双粒度协议

1.1 s2fs——一个可扩展的单一映像文件系统

s2fs是基于AIX操作系统实现的全局文件系统,它具备了严格的单一映像功能,实现了unix文件共享语义,并在不改动AIX核心的前提下,保证了与unix应用程序的完全二进制兼容. 另外,s2fs为了获得较好的可扩展性和I/O性能,它还实现了合作式缓存、并行分布式的元数据管理及并行存储功能.

s2fs由核心相关层和主体两部分组成. s2fs的核心层是在虚拟文件系统一级中实现的,它主要是接收来自逻辑文件系统的I/O请求. s2fs的用户层由3类用户级daemon构成,它们协调工作,共同完成与s2fs相关的I/O操作. 这3类daemon被分别称为客户(client)、管理器(manager)和存储器(storage). 应用程序发出与s2fs相关的系统调用,经由逻辑文件系统和虚拟文件系统,I/O请求传给s2fs的本地客户,若I/O请求不能由本地客户得到满足,则转发给管理器,由管理器通知相应的客户或存储器来完成具体的文件操作,把结果转发给本地客

---

* 本文研究得到国家自然科学基金和国家863高科技项目基金资助. 作者**王建勇**,1969年生,博士生,主要研究领域为并行/分布处理. **祝明发**,1945年生,博士,研究员,博士生导师,主要研究领域为高性能计算机系统与网络. **徐志伟**,1956年生,博士,研究员,博士生导师,主要研究领域为并行计算机系统.

本文通讯联系人:王建勇,北京100871,北京大学计算机科学与技术系网络分布与系统研究室

本文1998-07-20收到原稿,1998-09-07收到修改稿

户,然后经过虚拟文件系统和逻辑文件系统把结果返回给应用程序.图1给出了s2fs系统的组织结构.

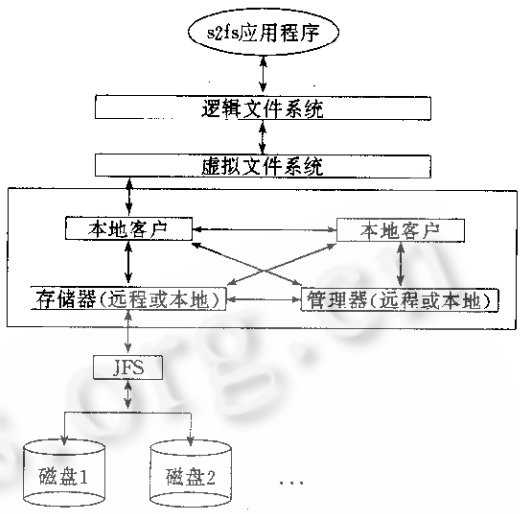

图1　s2fs系统透视图

1.2 双粒度协议

在分布式文件系统中,粗粒度的缓存一致性协议容易造成数据读写的"假共享"问题,而在细粒度一级来维护缓存一致性,则会给服务器带来太多的负担,并产生过多的不必要的开销.在粗粒度和细粒度两个级别上维护缓存一致性就可以弥补双方各自的不足之处,因而我们为s2fs设计并实现了双粒度协议.事实上,相对于单粒度协议而言,双粒度协议有下列好处:(1) 降低了因维护缓存一致性而引起的服务器端的工作负载;(2) 能够减少客户/服务器间的通信;(3) 更为重要的是,它能够提供某些hint信息,文件系统可以利用这些hint信息进一步缓和服务器的负担及网络开销.

s2fs是在块和文件两个粒度上维护缓存一致性的.与xFS类似,s2fs在块一级是利用读/写令牌来维护缓存一致性的,这里就不再赘述.在文件一级,s2fs利用回调(callback)机制来维护缓存一致性.当一个客户打开一个文件,并且不存在并发的共享写(concurrent write-sharing)时,管理器就赋予该客户一个被打开文件的回调,并允诺当其他客户以冲突方式打开该文件时,它会通知该客户放弃回调(callback break).当由于其他客户关闭某个以写方式打开的文件而使该文件不再处于写共享状态时,管理器又会重新赋予目前正处于打开该文件状态的客户一个回调.在实现s2fs的双粒度协议时,回调状态是存放在一个回调结构中的,在该结构中还记录了文件的索引号、目前正处于打开该文件状态的客户数目、文件长度及文件在磁盘上的位置(如存储分组的组号及起始节点号)等.这些信息不仅能够降低块的定位(block lookup),维护缓存一致性的开销,而且可以作为hint用于缓存替换,以提高I/O操作的性能.

2 启发式缓存替换算法

2.1 相关研究

合作式缓存为文件系统的存储层次引入了一个新的层次,即远程客户内存.不同的客户缓存算法以不同的方式管理这一新的存储层次.Daulin M. D.等人在文献[3]中阐述了几种主要的合作式缓存算法,其中N-chance转发算法是最为现实可行的、也是性能最好的合作式缓存算法.该算法根据客户的I/O行为动态地调整被合作式管理的那部分缓存的大小.它的主要思想是,优先缓存孤本缓存块(即整个系统的缓存中只有这样一个副本).当一个客户替换一个孤本时不是直接把它扔掉,而是随机地选择另一个客户,把该孤本转发给它,并且允许该孤本最多有N次转发机会.在实现N-chance算法时,通常要附加一些限制措施以防止"涟漪"效应,即被转发的孤本又引起另一个孤本的转发等等.

2.2 启发式缓存替换算法

虽然N-chance算法利用孤本优先策略,减少了系统缓存中同一块缓存的拷贝数目,相对地扩大了系统有效缓存的空间,但它仍有不足之处:(1) 每次发生缓存替换时,若被替换块不是其他客户转发过来的孤本,客户都要询问管理器,以判断被替换块是否为孤本,因而增加了网络开销和替换的响应时间;(2) 在转发孤本缓存块时,此算法是随机地选择将要接收孤本的目的客户,没有考虑到系统中各个客户的存取模式(如近期存取被转发孤本的概率)及状态(如空闲程度);(3) 不加区别地转发所有的孤本.事实上,有些孤本缓存块在近期甚至将永远不会被存取,对这些孤本的转发变得毫无意义.

我们设计的启发式缓存替换算法(以下简称为启发式算法)维护并利用某些hint信息,克服了N-chance算法的上述缺点,是一种增强型的N-chance算法.启发式算法所基于的hint信息有的来自双粒度协议,如在某客

户端某个文件是否处于打开状态、是否处于共享打开状态等等，这些 hint 信息不需要启发式算法做额外的工作. 还有一类 hint 信息是关于缓存块是否为孤本的，由客户和管理器共同来维护. 在实现启发式算法时，我们为客户端的缓存状态表增加了一项——有效缓存块的孤本状态标志 ncbc(number of cache block copies)，其值为 1 时，表示为孤本；其值为 2 时，表示系统中有该缓存块的多个缓存拷贝. 由于客户端进行缓存替换时只关心被替换块是否为孤本，因而 *ncbc* 标志就足以满足缓存替换的要求，至于系统中某缓存块的具体拷贝数目则是由管理器为维护缓存一致性而记录的. *ncbc* 的具体维护方法如下：

* 某缓存块第 1 次(从存储器的磁盘上)读/写到本地缓存中时，其 *ncbc* 置为孤本状态(即 $ncbc=1$)；

* 当客户之间转发数据块时，双方各自的 *ncbc* 置为 2，表示系统中有多个(≥ 2)拷贝；

* 当一个客户执行完写操作后，管理器将被写数据块在其他客户端的缓存块拷贝变为无效，同时，本地的被写数据块变为孤本状态($ncbc=1$)；

* 当由于某个客户执行缓存替换，使得整个系统中某块缓存的拷贝数目由 2 变为 1 时，管理器通知拥有该孤本缓存块的客户，使该数据块的 *ncbc* 置为孤本状态($ncbc=1$).

假设某一时刻，系统中的第 i 个客户的第 j 块被选为被替换对象，我们把它标记为 b_{ij}，并将 b_{ij} 所属的文件标记为 f，则此时 b_{ij} 所处的状态可能有：

A：b_{ij} 是由其他客户转发来的孤本；

A.1：b_{ij} 尚未被转发 N 次；

A.2：b_{ij} 已被转发 N 次；

B：目前无任何客户处于打开 f 的状态，这意味着 b_{ij} 近期(甚至永远将)不会被引用；

C：f 正处于打开状态；

C.1：只有本地客户打开了 f；

C.2：系统中有 $k(k\geq 2)$ 个客户处于打开 f 的状态；

C.2.1：b_{ij} 的 *ncbc* 等于 1；

C.2.2：b_{ij} 的 *ncbc* 等于 2.

在我们的启发式算法中，当 b_{ij} 处于 A.2，B 或 C.2.2 这 3 种状态之一时，无需转发，而是直接把 b_{ij} 丢弃；当 b_{ij} 处于状态 A.1 或 C.1 时，客户 i 随机地选择另一个客户 X，把 b_{ij} 转发给 X；当 b_{ij} 处于 C.2.1 状态时，客户 i 首先告诉管理器它目前要转发替换块 b_{ij}，管理器修改完缓存状态表后向客户 i 发送应答消息，应答消息中包含了将要接收 b_{ij} 的目的客户 Y，其中 Y 为一个正在使用 f 的客户(即处于打开 f 的状态). 客户 i 收到应答后把 b_{ij} 转发给客户 Y. 由于启发式算法基于 hint 信息，客户自己就能判断出 b_{ij} 是否为孤本，无需与管理器联系，因而同 N-chance 算法相比，在 B 及 C 状态下，启发式算法能够节省一对网络传输，又由于启发式算法能够根据客户本地的回调结构来判断被替换文件目前是否有客户正在使用，若没有客户使用(即处于 B 状态)，即使 b_{ij} 为孤本也不转发，因而使得系统中合作式缓存可以用来存放更多的有效数据块. 另外，虽然在 C.2.1 状态下需要征求管理器的意见，浪费了一对网络传输，但却增加了该转发后孤本的本地缓存命中率.

3 分析模型

为了评价启发式算法的性能，我们为启发式算法及 N-chance 算法建立了分析模型，以便对二者的性能进行比较. 为了简单起见，我们的分析模型只针对读操作的响应时间，这样做是合理的，因为系统中读操作所占的比例较大，是影响系统性能的主要因素.

假设一次读操作在不同的存储层次可得到满足的概率分别为：本地缓存命中率 P_l，远程命中率 P_r 及磁盘命中率 P_d，请求在不同存储层次得到满足的响应时间分别为 T_l，T_r 和 T_d，缓存替换所用时间为 $T_{替换}$，读操作的平均响应时间为 $T_{响应}$，则下列公式成立.

$$P_l+P_r+P_d=1, \tag{1}$$

$$T_{响应}=P_l\times T_l+P_r\times T_r+P_d\times T_d+(1-P_l)\times T_{替换}. \tag{2}$$

下面是缓存替换时影响性能的几个主要参数.

R_{ct}：合作式缓存(即用于存放其他客户转发来的孤本的缓存)占整个客户端缓存的比例；

R_{nl}：近期不会被引用的缓存块占本地缓存(即不包含合作式缓存)的比例；

P_{so}：文件处于共享打开状态的概率；

P_{sgl}：本地缓存中处于共享打开状态的文件的缓存块是孤本(即其 $ncbc=1$)的概率；

P_{h2}：本地客户处于打开 f 的状态，且系统中有且仅有两个客户处于打开 f 状态的概率；

P_{sr}：数据块共享读的概率，即客户 i 把孤本 b_{ij} 转发给客户 Y(其中客户 i 和 Y 都处于打开 f 的状态)后，又被客户 Y 本地读命中的概率；

N：N-chance 算法下远程缓存命中率 P_r 与本地缓存命中率 P_l 的比值，即下式成立，

$$N=\frac{P_r}{P_l}. \tag{3}$$

当在 N-chance 算法下发生缓存替换，并且被替换块 b_{ij} 不是由其他客户转发来的孤本时，本地客户都要向管理器发送消息以询问 b_{ij} 是否为孤本，因而共需要两次请求/应答网络传输. 假设一次网络传输的时间(这里指的是消息时延)为 T_{hop}，则 N-chance 算法下的替换开销为

$$T_{N\text{替换}}=(1-R_{ct})\times(2\times T_{hop}). \tag{4}$$

将式(1)、(3)及(4)代入式(2)后得到 N-chance 算法下的1次读操作的响应时间为

$$T_{N\text{-chance}}=(T_l-T_d+N\times(T_r-T_d)-2\times(1-R_{ct})\times T_{hop})\times P_l+T_d+2\times(1-R_{ct})\times T_{hop}. \tag{5}$$

我们用 P_A，P_B，P_C，P_{C1}，P_{C2}，P_{C21} 和 P_{C22} 分别表示缓存替换时被替换块处于状态 A，B，C，C.1，C.2，C.2.1 和 C.2.2 的概率，根据 2.2 节中被替换块状态及本节中影响性能的主要参数的定义，下列公式成立，

$$P_B+P_C=1-R_{ct}, \tag{6}$$

$$P_C=(P_B+P_C)\times(1-R_{nl})=(1-R_{ct})\times(1-R_{nl}), \tag{7}$$

$$P_{C1}=P_C\times(1-P_{so}), \tag{8}$$

$$P_{C2}=P_C\times P_{so}, \tag{9}$$

$$P_{C21}=P_{C2}\times P_{sgl}, \tag{10}$$

$$P_{C22}=P_{C2}\times(1-P_{sgl}). \tag{11}$$

当在启发式算法下进行缓存替换且 b_{ij} 处于状态 C.2.1 时，会引起两次网络传输，而处于 C.2.2 状态且在 $k=2$ 时，管理器应通知某个客户使其缓存块变为孤本，引起1次网络传输，故在启发式算法下的替换开销为

$$T_{\text{替换}}=P_{C21}\times 2\times T_{hop}-P_{C22}\times P_{k2}\times T_{hop}. \tag{12}$$

令 $P_{l\text{启}}$ 和 $P_{r\text{启}}$ 分别为启发式算法下的本地及远程缓存命中率. 同 N-chance 算法相比，启发式算法在状态 C.2.1下转发孤本时能够增加本地命中率，但不会改变总的缓存命中率(包括本地和远程)，所以下列公式成立.

$$P_{l\text{启}}+P_{r\text{启}}=P_l+P_r, \tag{13}$$

$$P_{l\text{启}}=P_l+P_{C21}\times P_{sr}. \tag{14}$$

将式(3)、式(6)～(14)代入式(2)后可得到

$$T_{\text{启发}}=T_{N\text{-chance}}+(T_l-T_r)\times(1-P_l)\times(1-R_{ct})\times(1-R_{nl})\times P_{so}\times P_{sgl}\times P_{sr}+(1-P_l)\times(1-R_{ct})\times T_{hop}\times$$
$$((1-(1-R_{ct})\times(1-R_{nl})\times P_{so}\times P_{sgl}\times P_{sr})\times(1-R_{nl})\times P_{so}\times(2\times P_{sgl}+(1-P_{sgl})\times P_{h2})-2). \tag{15}$$

4　性能评价

由于在 s2fs 系统所基于的平台中，各个节点之间是用 10Mbit/s 以太网互连的，因而我们关于系统配置的某些基本假设取自文献[3]中关于以太网的数据：缓存块的大小为 8Kbytes，本地内存的存取时间为 250μs，网络时延 $T_{hop}=200\mu s$，单块数据的传输时间为 6 250μs，从磁盘读取一块数据的时间为 14 800μs，那么，$T_l=0.25$ms. 与文献[3]不同的是，本文中的 $T_r=7.1$ms，$T_d=21.9$ms. 因为在我们的系统中元数据的管理与数据的存储和缓存是分开的，从远程缓存或磁盘上存取数据要用 3 次网络传输.

对于影响读操作响应时间的参数的取值我们是这样处理的：对每个参数都取 3 组值，即下限值、标准值和上

限值.参数 R_{ct},P_l,N,R_{nl},P_{so}及 P_{k2}是从文献[4,5]分析得来的.另外,P_l 和 N 是一块取值的,因为二者是相关的,当 P_l 变大时,N 往往会变小;当 P_l 变小时,N 往往会变大.对于 P_{sr}和 P_{sgl},我们分别取 10%,50%和 90%作为其下限值、标准值及上限值.这些影响性能的主要参数的取值情况见表 1.

表 1 某些影响系统性能的参数取值范围

	R_{ct}	P_l,N	R_{nl}	P_{so}	P_{k2}	P_{sr}	P_{sgl}
下限值	0.05	0.67,0.212	0.05	0.40	0.02	0.10	0.10
标准值	0.30	0.78,0.18	0.15	0.65	0.06	0.50	0.50
上限值	0.55	0.94,0.04	0.25	0.90	0.10	0.90	0.90

我们将参数 R_{ct},P_l,N,R_{nl},P_{so},P_{k2},P_{sr}和 P_{sgl}不同取值的组合代入公式(5)和(15),可以分别得到在 N-chance 算法和启发式算法下,一次读操作的响应时间.分析结果表明,当这些参数取各种不同的值时,读操作的响应时间在启发式算法下比在 N-chance 算法下都有不同程度的降低.例如,当这些参数都取标准值时,启发式算法相对于 N-chance 算法,其读操作的响应时间能够降低 6.4%;当 R_{ct},R_{nl}及 P_{k2}取下限值,P_l,N 取标准值,P_{so},P_{sr}和 P_{sgl}取上限值时,在启发式算法下,读操作的响应时间降低了 34.885%;即使在最坏情况下,即 R_{ct},P_l,R_{nl}及 P_{k2}取上限值,P_{so},N,P_{sr}和 P_{sgl}取下限值时,启发式算法相对于 N-chance 算法,也能够使读操作的响应时间降低 1.085%.图 2 即说明了这一点.

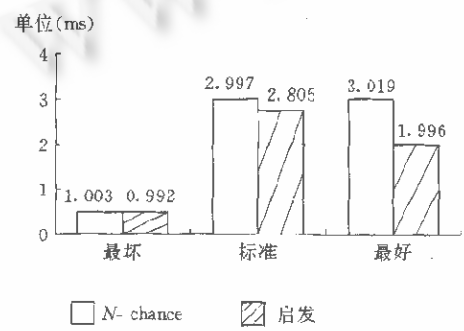

图2 两种算法的性能比较

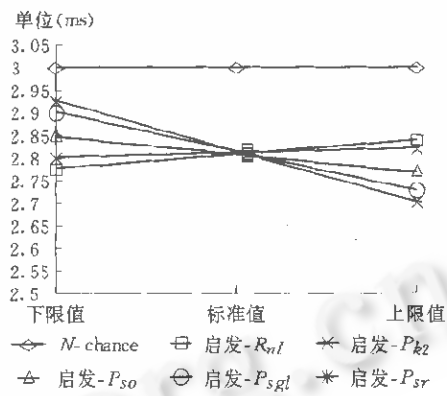

图3 R_{nl},P_{so},P_{k2},R_{sr}及R_{sgl}的取值对启发式算法的影响

图 3 考察了当 R_{ct},P_l 及 N 取标准值,R_{nl},P_{so},P_{k2},P_{sr}和 P_{sgl}取不同的值时对启发式算法的影响.从图 3 中可以看出,P_{so},P_{sr}和 P_{sgl}取值越大,启发式算法对读操作响应时间的改进也越大.反之,R_{nl}和 P_{k2}取值越大,启发式算法对读操作响应时间的改进越小,但影响程度不大.另外,图 3 也说明了不论 R_{nl},P_{so},P_{k2},P_{sr}和 P_{sgl}取什么值,在降低 I/O 的响应时间方面,启发式算法都优于 N-chance 算法.

5 结 论

本文在 s2fs 系统的双粒度协议基础上,提出了一种基于 hint 的启发式合作缓存替换算法,用分析模型说明了启发式算法比现有的 N-chance 算法在降低 I/O 的响应时间方面有进一步的改善.今后,我们将在 s2fs 系统中具体实现启发式算法,并用实际的应用来测试启发式算法对 I/O 性能的改进.

参考文献

1 Leff Avraham, Wolf Joel L, Yu Philip S. Replication algorithms in a remote caching architecture. IEEE Transactions on Parallel and Distributed Systems, 1993,4(11):1185~1204

2 Anderson T E *et al*. Serverless network file systems. ACM Transactions on Computer Systems, 1996,14(1):41~79

3 Dahlin M D *et al*. Cooperative caching: using remote client memory to improve file system performance. In: Proceedings of the 1st Symposium on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 1994. 276~280

4 Baker M G *et al*. Measurements of a distributed file system. ACM Operating Systems Review, 1991,25(5):198~212
5 Blaze M A. Caching in large-scale distributed file systems [Ph. D. Thesis]. Department of Computer Science, Princeton University, 1993

Hint-based Heuristic Cache Replacement Algorithm under Dual-granularity Protocol

WANG Jian-yong[1,3]　ZHU Ming-fa[2,3]　XU Zhi-wei[2,3]

[1](*Department of Computer Science and Technology　Beijing University　Beijing*　100871)
[2](*National Research Center for Intelligent Computing Systems　Beijing*　100080)
[3](*Institute of Computing Technology　The Chinese Academy of Sciences　Beijing*　100080)

Abstract　Cooperative caching is one of the key technologies used to improve the performance of a cluster file system. s2fs (scalable single-image file system), a single-image cluster file system prototype, uses dual-granularity cache coherence protocol in order to implement efficient cooperative caching which meets the needs for strict UNIX-semantics. In this paper, a hint-based heuristic cache replacement algorithm under s2fs' dual-granularity protocol is proposed, and the analytical models are established for heuristic algorithm and one of the existing coordinated algorithm —— *N*-chance. The analytical results show that the heuristic algorithm can effectively reduce the I/O response time compared with *N*-chance algorithm almost in each case.

Key words　Heuristic cache replacement algorithm, *N*-chance algorithm, dual-granularity protocol, s2fs (scalable single-image file system), cooperative caching.