一种适应WAN的新型分层缓存管理机制

卢 军*[1] 陈凌虎[2] 卢显良[1]

(1. 电子科技大学计算机学院 成都 610054; 2. 信息产业部经济体制改革与经济运行司 北京 100846)

【摘要】提出了一种新型的分层Cache管理机制。该机制是针对WAN环境中大型分布式文件系统设计，完全适应在WAN中工作。HCMM使用分层的方法来管理WAN中的Cache，可以有效地减少WAN上的网络通信开销和服务器上管理Cache的开销，从而提高WAN中分布式文件系统的性能。建立了HCMM的理论分析模型，并和单层Cache管理机制进行了性能对比，结果表明，HCMM在WAN环境中比单层Cache管理机制的性能更好。

关 键 词 缓存; 广域网; 分层; 性能

中图分类号 TP316.4

A Novel Hierarchical Cache Consistence Mechanism for WAN

Lu Jun[1] Chen Linghu[2] Lu Xianliang[1]

(1. College of Computer Science and Engincering, UEST of China Chengdu 610054;
2. Department of ESR & EO, Ministry of Information Industry of China Beijin 100846)

Abstract This paper presents a novel hierarchical cache consistence mechanism–HCMM. HCMM is specially designed for the large scale distributed file system in WAN and can well adapt to work in WAN. HCMM uses hierarchical method to manage the consistence of cache, and can minimize the overhead of network communication and cache management. This paper also sets up a theory model to analyze the performance of HCMM. The results of theory analysis prove that the performance of HCMM is better than conventional cache consistence mechanism.

Key words cache; wide area network; hierarchical; performance

目前，分布式文件系统向着越来越大的规模发展，同时希望建立一种分布式文件系统来管理更广范围的文件资源，例如管理WAN环境中的文件资源，或管理Web环境中的文件资源。缓存(Cache)管理机制是影响分布式文件系统性能和可扩展性的重要因素[1]。传统分布式文件系统中使用的Cache管理机制大多是针对高带宽、低时延的LAN设计，不能适应低带宽、高时延的WAN环境[2]。

Coda是Carnegie Mellon University开发的分布式文件系统[3]，它通过服务器执行Call-back来实现Cache一致性管理。虽然Coda采用了AVSG(Access Volume Storage Group)机制来提高Call-back的执行效率，但因Coda的体系结构是单层的Client/Server结构，所有的Call-back工作必须由少数服务器担任，因此，随着系统规模的增加，文件服务器将成为系统性能的瓶颈。

本文提出了一种新型的分层Cache管理机制(HCMM)。HCMM针对WAN环境低带宽、高时延的特性，采用了分层的Cache管理机制，Cache管理工作由多个服务器分担。HCMM可以有效地减少WAN上大型分布式文件系统的网络通信开销和服务器管理Cache的开销来提供更高的性能。对HCMM的性能进行了理论分析和对比试验，结果表明，在WAN环境中HCMM比传统的单层Cache

2002年4月2日收稿
* 男 28岁 博士生

管理机制性能更好。

1 HCMM中的分层结构

HCMM的体系结构如图1所示，HCMM由若干域(Domain)组成，域内通过高速LAN连接，域间是低速WAN连接。DCS(Domain Cache Server)担负每个域中的Cache管理任务，MCS(Main Cache Server)担负所有DCS的Cache管理任务。

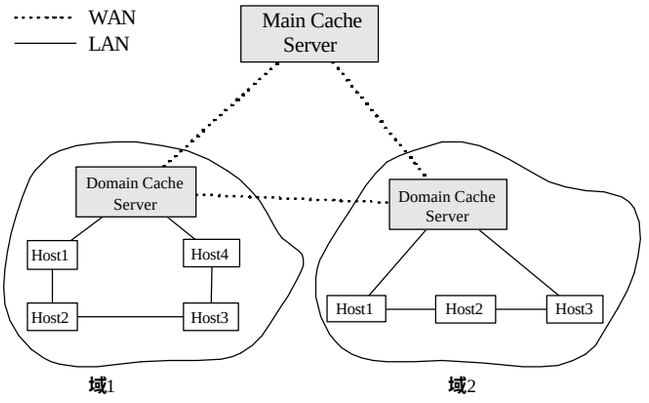

图1 HCMM体系结构图

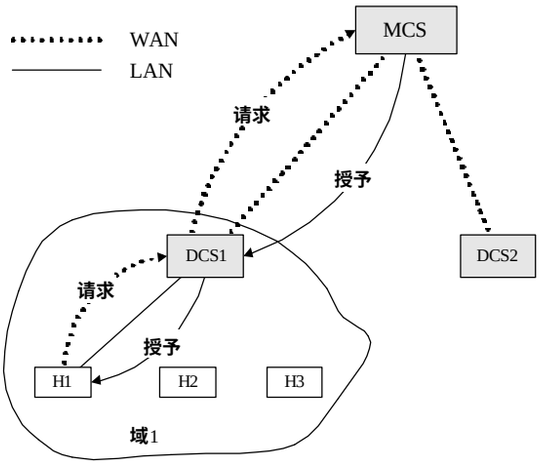

图2 HCMM中的分级锁管理机制

在HCMM中每一个文件都有读锁(Reading Lock)和写锁(Writing Lock)，只有获得了读锁或写锁的客户才能对文件进行读或写操作。一个拥有读锁的客户可以在本地对文件内容进行缓存，供以后多次读取；拥有写锁的客户可以独享地在本地缓存中对文件进行任意读写操作。服务器既可以向客户分发读锁和写锁，也可以要求客户释放读锁和写锁。当一个客户释放读锁时，必须使自己的读缓存无效；当客户释放写锁时，必须将自己本地缓存中的脏数据刷新到服务器。

在HCMM中使用多读/单写协议来保证Cache的一致性[4]，即多个客户可以同时拥有同一个文件的读锁，但是在任何时刻只能有一个客户拥有一个文件的写锁。如图2所示，HCMM使用了两层文件锁管理机制。如果在Domain1中主机H1要申请文件F1的读锁，H1首先向DCS1申请。这里可能有两种情况：1) 如果DCS1已经拥有了F1的读锁，DCS1就直接将读锁赋予H1，而不用到MCS申请；2) 如果DCS1没有F1的读锁，DCS1就将向MCS申请读锁。MCS收到DCS1的申请后，也有两种情况：(1) MCS发现没有其他DCS拥有F1的写锁，MCS可以将F1的读锁赋予DCS1；(2) MCS发现有其他DCS(设为DCS2)拥有F1的写锁，这时MCS必须先请求DCS2释放F1的写锁，然后才可以将F1的读锁赋予DCS1。当DCS1获得F1的读锁后，再将读锁赋予H1。

如果在Domain1中主机H2申请文件F2的写锁，H2首先向DCS1发出申请，这时也有两种情况：1) 如果DCS1拥有写锁，表示此域中已有一台主机(设为H3)拥有了该文件的写锁。根据多读/单写协议，DCS1必须首先要求H3释放写锁，然后才能将写锁赋予H2。DCS1因此向H3发送消息要求释放F2的写锁，当H3释放写锁后，DCS1将F2的写锁赋予H2；2) 如果DCS1没有拥有F2的写锁，DCS1将向MCS申请F2的写锁，此时又将出现两种情况：(1) MCS发现没有任何DCS拥有F2的写锁，这时MCS可以直接将F2的写锁赋予DCS1，DCS1再将F2的写锁赋予H2；(2) MCS发现有一个DCS(设为DCS2)拥有F2的写锁，那么MCS必须向DCS2发送消息请求释放F2的写锁。当DCS2释放写锁后，MCS才能将F2的写锁赋予DCS1，DCS1再将F2的写锁赋予H2。

HCMM分层体系结构的优点如下：1) 整个系统的Cache管理工作由DCS和MCS分担，避免了Cache管理任务集中在少数服务器上，而使这些服务器成为系统扩展的瓶颈[5]；2) 在HCMM中域内

通过LAN连接，带宽大、延迟小，大量的Cache管理工作在域内就可以完成，无需通过WAN和MCS通信，这样减少了WAN上的通信量，提高了系统的性能。

2 HCMM性能分析

设N代表HCMM中所有主机的数量，R代表单位时间内文件读访问率，W代表单位时间内文件写访问率。

HCMM中的分级锁申请/释放机制如图3所示，图中，横坐标代表时间t，N个客户独立地以泊松分布发出读请求和写请求，读请求的概率为R，写请求的概率为W[6]。T周期代表客户没有拥有锁，L周期代表客户拥有读锁。一个T周期在读锁释放的时候开始，在下一次文件读的时候结束，因此T周期的长度是$T_T = \frac{1}{R}$。

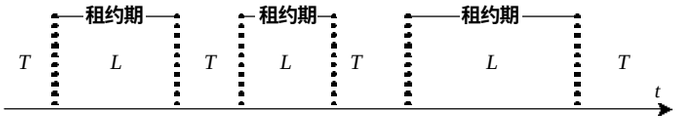

图3　HCMM中的分级锁申请/释放的机制

当客户接收到一个读锁时，L周期开始，在下一次写申请开始的时候结束。假设有一个Poisson过程以概率为NW运行，代表读锁被写锁申请取消，所以L周期的长度为

$$T_L = \frac{1}{NW}$$

因此，一个客户拥有读锁的时间为

$$T_{RL} = \frac{T_L}{T_L + T_T} = \frac{R}{NW + R}$$

由于N个客户独立地访问文件，因此同时拥有一个文件读锁的客户数量是

$$N_L = NT_{RL} = \frac{NR}{NW + R}$$

在HCMM中Cache管理机制分为两层，各层之间的连接情况如图4所示，其中$D_{proc(H)}$、$D_{proc(D)}$、$D_{proc(M)}$分别代表在主机、域服务器和主服务器上处理一个锁请求的时间。$D_{prop(L)}$、$D_{prop(W)}$分别代表在LAN和WAN上传输消息的时延。假设MCS向Host发送一个消息，这个过程的时延是

$$D_{send} = D_{proc(M)} + D_{prop(W)} + D_{proc(D)} + D_{proc(D)} + D_{prop(L)} + D_{proc(H)}$$

假设$D_{proc(M)} = D_{proc(D)} = D_{proc(H)} = D_{proc}$，则$D_{send} = 4D_{proc} + D_{prop(W)} + D_{prop(L)}$。同样，Host向MCS回复消息的时延为$D_{reply} = 4D_{proc} + D_{prop(W)} + D_{prop(L)}$。因此，MCS向Host发送消息得到并处理Host回复消息的时延为

$$D_{send\ and\ reply} = D_{send} + D_{reply} = 2(4D_{proc} + D_{prop(W)} + D_{prop(L)})$$

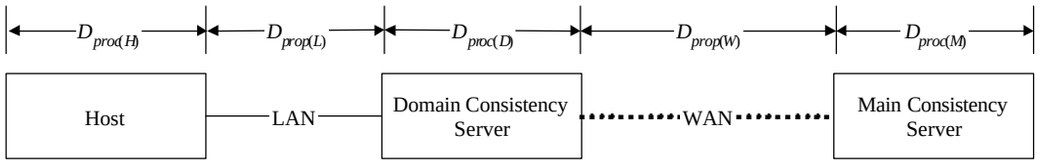

图4　分级汇总Lease处理时延

在一个域中，一个主机对文件F发出读请求，设DCS拥有F文件读锁的概率为P。在文件读锁有效时间内，客户可以通过拥有读锁来减少向服务器发送读请求的数量，设减少的比例为R_{lc}，则在单位时间内服务器处理的读请求的数量$2NR$应该除以$(1+R_{lc})$，故MCS服务器每个读请求的平均处理的消息数量(负荷)为

$$L_R = \frac{(1-P)(2NR)}{1+R_{tc}}$$

假设在HCMM中每个域的情况一样，因此其他各域的L_R相同，每个读请求的平均延迟时间为

$$D_R = \frac{(1-P)}{1+R_{tc}}(D_{send}+D_{replay}) = \frac{(1-P)[2\times(4D_{proc}+D_{prop(W)}+D_{prop(L)})]+2P(D_{prop(L)}+2D_{proc})}{1+R_{tc}}$$

当一个客户对某个文件写请求的时候，MCS必须向所有拥有文件读锁的DCS发送消息请求释放读锁。设拥有读锁的DCS数量为N_{DCS}。每个DCS接收到消息后，都会向MCS发送返回消息，MCS必须对这些返回消息进行处理。由于MCS不必要求写请求的DCS释放读锁，因此要求释放的DCS数量N_R是

$$N_R = N_{\mathrm{DCS}} - \frac{N_{\mathrm{DCS}}}{N} = \frac{N_{\mathrm{DCS}}(N-1)}{N}$$

MCS处理这些请求的时延为

$$D_{send}+D_{replay}=[2D_{prop(W)}+(2N_R+2)\ D_{proc}]+\ [2D_{prop(L)}+4\ D_{proc}]$$

单位时间内，服务器处理的消息数量为

$$L_W = NW\left(\frac{N_{\mathrm{DCS}}(N-1)}{N}+\frac{N_{\mathrm{DCS}}(N-1)}{N}\right) = 2N_{\mathrm{DCS}}(N-1)W$$

每个写请求的平均延迟时间为

$$D_W = D_{send}+D_{replay} = [2D_{prop(W)}+(2N_R+2)D_{proc}]+[2D_{prop(L)}+4D_{proc}]$$

则单位时间内服务器处理读和写请求的消息负荷为

$$L_R + L_W = \frac{(1-P)(2NR)}{1+R_{tc}} + 2N_{\mathrm{DCS}}(N-1)W$$

服务器处理一次读或写的平均时延为

$$\frac{RD_r+WD_w}{W+R}$$

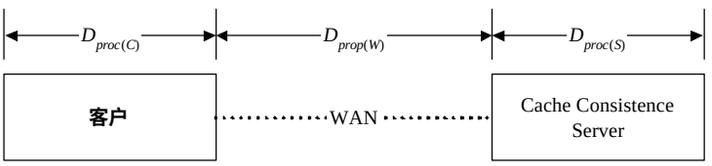

图5 服务器和客户之间的Lease处理时延

使用同样的分析方法，对图5所示的单层Cache管理机制，服务器在单位时间内处理读请求消息的数量为

$$L_R = \frac{2NR}{1+R_{tc}}$$

每个读请求的平均延迟时间为

$$D_r = \frac{2[2D_{proc}+D_{prop(W)}]}{1+R_{tc}}$$

其中

$$R_{tc} = RT_L = R\frac{1}{NW}$$

单位时间内，服务器处理的写请求消息数量为

$$L_W = NW\left(\frac{N_C(N-1)}{N}+\frac{N_C(N-1)}{N}\right) = 2N_C(N-1)W$$

每个写请求的平均时延为

$$D_w = 2D_{prop(W)} + (2N_C + 2)D_{proc}$$

因此，在单层模型下，单位时间内服务器处理读和写请求总消息处理数量为

$$L_R + L_W = \frac{2NR}{1+R_{tc}} + 2NWN_C$$

一次读或写的平均时延为

$$\frac{RD_r + WD_w}{W + R}$$

3　性能对比

在进行性能对比之前，先对第2节中描述的模型进行验证，实验使用9台计算机组成了类似图1所示的试验系统，不同的是每个域内都是3台主机。在域内使用10 M以太网连接，域间的消息传递时延设置为从0.005~0.500 s。在模拟试验中N =6，R =0.1，W=0.001，D_{proc}=0.010 s，$D_{prop(L)}$=0.005 s，$D_{prop(W)}$ =0.500 s，P=0.3，R_{tc}=0.2。在模拟试验中，测试了在不同的D_w/D_L值下的响应时间，并把响应时间和对应的模型计算结果进行了比较，比较结果如图6所示。从图6可以看到，本文的模型计算结果和测试结果吻合较好。测试结果比模型计算结果大的原因是在模型中没有考虑网络的动态变化延迟、集线器和路由器延迟等因素。

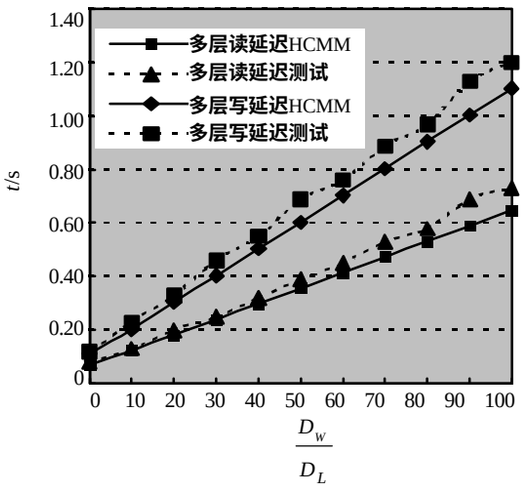

图6　模型验证结果

在对模型进行验证的基础上，本文用性能分析方法对HCMM和单层模型进行对比。HCMM中划分的域数量不同对读写响应时间的影响如图7、8所示。图中，N=1 000，R=0.1，W=0.001，D_{proc}=0.010 s，$D_{prop(L)}$=0.005 s，$D_{prop(W)}$ =0.500 s。从图7可以看到，HCMM的响应时间随着域个数的增加而增加，但是始终比单层模型低。当HCMM的域个数增加到一定程度时(图7中大约是N的1/10)，HCMM的读响应时间逼近于单层模型的响应时间。从图8可以看到，HCMM的写响应时间较单层模型都低，当域的

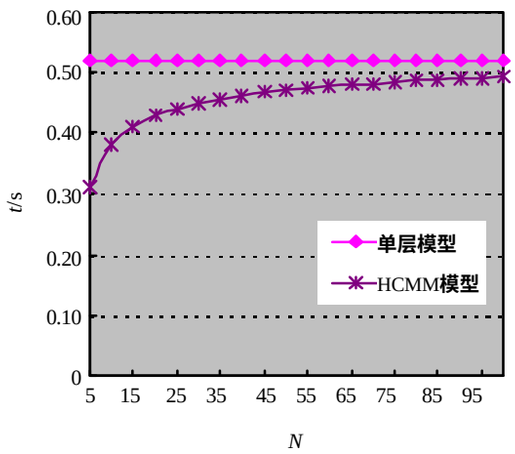

图7　域的个数和读请求响应时间比较

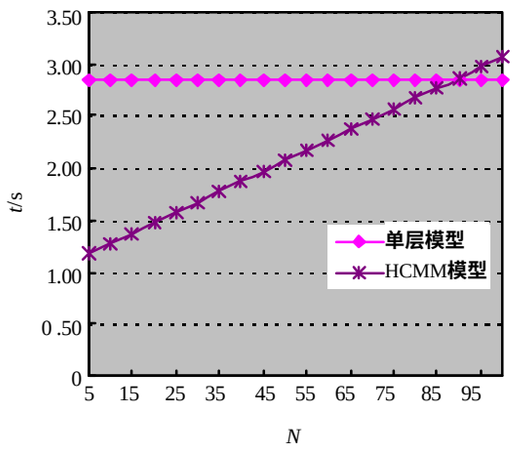

图8　域的个数和写请求响应时间比较

个数大于90时，HCMM的写响应时间才大于单层模型。这是因为域的个数增多的时候，HCMM中

写请求时，要求释放的DCS数目也增多了，这时HCMM的写响应时间才大于单层模型。由于HCMM中使用了分层域机制，读和写请求时间都比单层模型更低。

图9是不同网络速度下HCMM和单层模型读写速度的对比，图中N =1 000，N_{DCS} =10，R =0.1，W =0.001，D_{proc} =0.010 s，D_L =0.005 s。从图9可以看到，HCMM在不同的网络速度下，其读写性能比单层模型更好，而且随着WAN速度和LAN速度的差别加大，HCMM的性能优势将更明显。

图10显示了在不同读写概率下HCMM和单层模型处理消息的性能，图中W =1/1 000，N =1 000，N_{DCS} =10。从图10可以看到，当读概率大于写概率10倍以上时，HCMM的消息处理数量远远小于单层模型。这表明HCMM的读性能非常好，而且读越频繁，其性能就比单层模型更好。

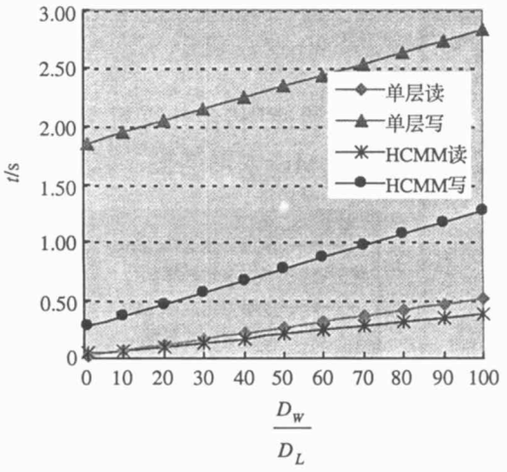

图9 不同网络速度对响应时间的影响

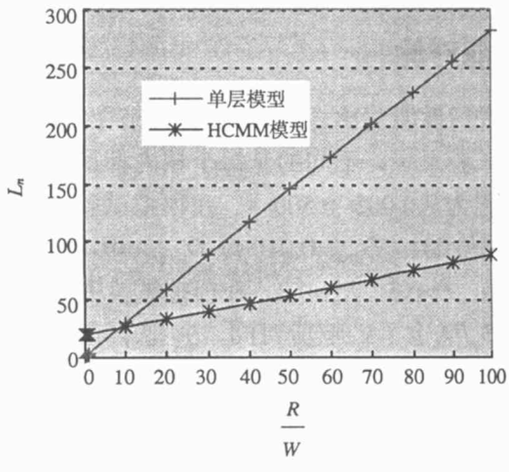

图10 读/写率对处理租约个数的影响

4 结 束 语

本文提出了一种新型分层Cache管理机制(HCMM)。HCMM使用分层体系结构，将WAN环境中的Cache管理工作分担到若干域中来完成，域内通过LAN连接，域间通过WAN连接。每个域的Cache管理工作由域服务器完成，所有的域服务器由主服务器管理。这种体系结构保证了HCMM具有很好的伸缩性和扩展性。HCMM有效地降低了服务器维护Cache一致性的负荷，避免了传统Cache管理机制中的服务器负荷瓶颈问题，非常适合WAN中的大型分布式文件系统。

参 考 文 献

1 Wang R, Anderson T, Dalin M. Experience with a distributed file system implementation. Technical Report, University of California, Berkeley, Computer Science Division, 1997

2 Blaze M, Alonso R. Toward massive distributed systems. Proceedings of the 3rd Workshop on Workstation Operating Systems, 1992, 48-51

3 Satyanarayananan M, Kistler J J, Kumar P, *et al.* Coda: A highly available file system for a distributed workstation. IEEE Transactions on Computers, 1990, 39(4): 245-254

4 Thekkath C, Mann T, Lee E. Frangipani: a scalable distributed file system. Proceedings of the 16th ACM Symposium on Operating Systems Principles, 1997, 224-237

5 Dahlin M, Mather M,Wang R, *et al.* A quantitative analysis of cache policies for scalable network file systems. In SIGMETRICS, 1994

6 刘 熠，蔡希尧. 面向网络的一种新分布式事务处理协议. 电子科技大学学报，1997, 26(2): 175-179