移动 Web 访问高性能缓存验证算法*

周 桓[1,2], 李 京[1], 冯玉琳[1,2]

[1](中国科学院软件研究所 计算机科学重点实验室,北京 100080);
[2](中国科学院软件研究所 软件工程技术中心,北京 100080)
E-mail: {quan,lij}@otcaix.iscas.ac.cn; feng@ios.ac.cn
http://www.ios.ac.cn

摘要: 提出一种基于客户-代理-服务器的卷回调算法来解决传统回调算法所面临的两个主要问题:可扩充性和客户端断连.代理可以分担服务器的负载,提高系统的可扩充性;基于卷的回调可以加速验证过程,消除客户从断连状态恢复时的重新验证瓶颈.试验结果表明,改进的卷回调算法可以减少 86%的验证消息,从而使得强一致性的回调算法在总体性能上已经接近通常使用的弱一致性的 ATTL(adaptive time to live)算法.

关 键 词: 缓存验证算法;回调;卷;移动计算;断连
中图法分类号: TP393 文献标识码: A

迅速发展的无线通信技术和便携式计算设备为网络分布计算提供了一个全新的广阔应用领域.数字无线通信网络作为对有线主干网络的一种扩充和延伸,使得各种便携式计算设备不再是孤立的计算实体,而是网络计算环境中的一部分.在移动通信环境中,个人可以通过各种数字辅助设备,使用各种不同的网络接入手段,随时随地访问共享信息,甚至在移动中访问信息.可以说,人们正由分布式计算进入移动计算时代.在各种信息共享系统中,WWW 具有瘦客户、一致的访问界面和巨大的信息库等特点,是一种很好的移动信息访问方式.

移动性打破了传统的分布式系统的一些假定,带来了新的约束[1].因此,人们需要重新研究分布式系统的模型和算法来解决这些约束.缓存是减少移动数据访问通信开销的主要途径.缓存验证算法为本地缓存中的数据提供一致性保证.由于无线网络上的回路延迟远远大于有线网络,同时移动客户端缓存的数据量比较大,因此,验证过程中的交互过程不仅造成了额外的通信开销,还延长了读操作的延迟.目前通常采用的解决方式是放松一致性限制,延长缓存对象的存活期[2].虽然这种途径可以降低通信开销,但用户读到脏数据的可能性大大增加了.而在理想的网络连接下,回调验证算法可以把维持强一致性的通信开销降低到最小,文献[3]着重比较了回调算法和 ATTL(adaptive time to live)算法在有线 WWW 环境下的性能,并论证了用回调算法来维持数据强一致性的可行性.

本文提出一种基于客户-代理-服务器的卷回调算法来解决上述问题.客户-代理-服务器结构是一种适合于无线网络系统的系统结构,可以很好地兼容现有的缓存验证协议.代理可以分担服务器的负载,提高系统的可扩充性.卷是缓存对象的逻辑组织,基于卷的大粒度验证可以加速客户从断连中恢复时的重新验证过程.试验结果表明,这种回调算法在没有过多增加通信开销的前提下保证了缓存的强一致性.

本文第 1 节探讨了移动性对缓存验证算法的影响.第 2 节描述了卷回调算法.第 3 节给出性能测试过程和结果.在第 4 节中,我们把本文的工作和其他相关工作进行了比较.第 5 节给出结论.

* 收稿日期: 2000-07-26; 修改日期: 2000-08-07
基金项目: 国家自然科学基金资助项目(69833030);国家重点基础研究发展规划 973 资助项目(G1998030400)
作者简介: 周桓(1976－),男,安徽芜湖人,博士,主要研究领域为移动计算,分布式系统;李京(1966－),男,江苏无锡人,博士,研究员,博士生导师,主要研究领域为分布对象计算,软件工程方法学,移动计算;冯玉琳(1942－),男,江苏泰县人,博士,研究员,博士生导师,主要研究领域为软件工程,形式化方法,分布对象计算.

1 移动环境对缓存验证算法的影响

在传统的分布式系统中,缓存的一致性可以通过以下两种方式得到:客户查询和服务器回调.在客户查询方式中,在客户端发生缓存命中时,缓存管理器需要首先向数据服务器发送消息查询本地缓存是否与服务器上的数据一致.如果是,则服务器返回确认信息;否则,服务器传回更新后的数据.在服务器回调方式中,服务器记录每条数据被哪些客户缓存了,在修改数据之前,服务器要通知所有缓存了被修改数据的客户,客户收到通知以后把失效的数据从缓存中删除,服务器只有在得到所有客户的确认消息以后才能真正完成修改操作.根据所提供的数据一致性保证不同,缓存验证算法又可以分为强一致性算法和弱一致性算法.在强一致性算法中,从缓存中读取的数据总是和数据源一致,在弱一致性算法中则允许读操作返回脏数据.其中,在客户询问方式中为了降低因每次询问服务器而造成的巨大通信开销和读延迟,人们提出了自适应存活期的缓存验证算法 ATTL.它根据缓存对象的"年龄"确定存活期,在存活期内该对象被认为是最新的,只有在存活期满后客户才向服务器查询其有效性.因此,这种算法是以牺牲数据一致性为代价来加速验证过程.

在移动计算系统中,服务器通常仍然位于固定的有线网络上,客户则可以是无线网络中的手持设备.移动客户上的缓存验证算法必须考虑以下几个因素:

· 数据一致性.弱一致性保证可以满足用户浏览的需要,但一些基于 Web 的应用,如 Agent 等,则需要提供数据的强一致性保证.

· 无线通信代价.由于有线和无线网络性能的差异,缓存验证算法需要着重考虑减少在无线网络上传输的消息量,从而降低系统的通信开销和操作延迟.

· 客户断连.移动设备的网络连接被断开是移动环境中经常出现的现象,而不像在传统的分布式系统中被看作是灾难性事件.缓存验证算法必须提供断连恢复手段,以支持移动计算系统的断连操作.

基于以上分析,表 1 列出了各种验证算法的性能指标.其中,对于缓存对象,R 是单位时间内读操作的次数,W 是单位时间内写操作的次数;Tr 是网络回路延迟;Tt 是数据传输时间;N 是缓存中对象的个数;V 是卷的个数.

Table 1 Comparison of cache consistency algorithms

表 **1** 缓存验证算法对比

	Poll each time①	ATTL	Callback②	Volume③
Consistency④	Strong⑨	Weak⑩	Strong	Strong
Messages per read⑤	1	$\mathrm{Min}(1/(R*TTL),1)$	0	0
Read latency⑥	$Tr+1/W*Tt$	$\le\min(1/(R*TTL),1)*Tr+1/W*Tt$	0	0
Messages per write⑦	0	0	1	1
Revalidation messages⑧	0	0	N	$V+\sum failed(v)Size(v)$

①每次查询,②回调,③卷回调,④数据一致性,⑤每次读的消息数,⑥读延迟,
⑦每次写的消息数,⑧每次断连后的重新验证消息,⑨强,⑩弱.

2 算法设计

在基于卷的回调算法中,数据对象按照逻辑关系分成若干组,每一组称为一个卷(volume).卷回调算法提高性能的主要原因是,在一个作业期内只有很少一部分访问过的对象会被修改.因此,通过增大验证过程的粒度,卷回调算法可以消除因客户断连所造成的重新验证过程瓶颈.

本节首先介绍 Web 环境下的基本卷回调缓存验证算法,然后分节讨论可扩充性、断连操作和动态卷调整.

2.1 基本卷回调算法

在基本的卷回调算法中,对象是指被客户缓存的单个 Web 文件,除了数据内容和 URL(uniform resource locator)以外,对象的属性还包括上次修改时间以验证对象是否有效.卷是一组对象的集合,它们来自同一个 Web 服务器.卷的有效性是通过其版本号来验证的,每次对卷中添加或删除一个对象都会导致版本号的增加,因此,卷上的回调实际上是建立在卷版本号之上的.

在传统的缓存对象信息之外,客户还要维护一个卷信息表,服务器则要为每个客户维护一个卷信息表,表中包括卷所包含的对象和版本号.当服务器收到客户新的请求时,在响应前要为该客户建立对象和卷的状态.客户

收到响应后也需要把该对象添加到本地卷状态表中.如果客户以前没有访问过该服务器,则需要创建相应的卷.

服务器在修改文件之前,首先把对象从它所在的卷中删除并更新卷的版本号,然后通知客户.客户收到通知后也相应地把本地缓存对象从对应的卷中删除,并同步更新卷版本号.只有在收到所有缓存了将被修改的文件的客户的确认消息以后,服务器才能完成写操作.因此,对于每个缓存数据项,本算法同时维护了两个回调:对象粒度的回调和卷粒度的回调,其中卷粒度的回调保证了卷中所有对象的有效性.只要其中任何一个回调没有打破,客户缓存管理器就可以直接返回缓存数据而不需要向服务器查询对象的有效性.

2.2 可扩充性

传统回调算法的一个重要缺陷就是可扩充性差.尤其在 WWW 这样的广域网络环境中的信息共享系统中,Web 服务器维持客户缓存信息的代价是很高的;另一方面,等待所有客户返回确认信息大大增加了写操作的延迟.为此,我们在客户和服务器之间增加一层代理来增强系统的可扩充性.

代理已经广泛地应用于WWW中,HTTP/1.1中定义了缓存代理的概念,它为一组客户提供共享的缓存空间.同时,在无线网络体系结构中,无线网络通常要通过接入点连接到有线主干网络上,在诸如 Mobile IP 和 Indirect TCP 的无线网络的 Internet 协议中,也通常把接入点看成具有不同功能的代理.因此,在缓存验证算法中加入代理层是一种自然的扩充,不需要额外的软、硬件支持.

对于 Web 服务器,代理代表了它所服务的一组客户;对于客户,代理代表了所有的 Web 服务器.它除了转发请求和响应之外,还要转发验证消息和修改通知.代理要为每个它所服务的客户维持一份卷状态表.由于有线网络上的通信代价不是本文讨论的内容,为简化实现,我们在代理和服务器之间采用传统的回调算法.服务器只需为每个代理保存对象缓存状态表,从而减少了状态表的大小以及服务器等待客户确认修改的时间.

代理和客户之间使用上述卷回调算法传送数据修改信息和同步修改卷信息表.客户断连会导致它们之间卷状态表不一致,客户从断连恢复时会检测到不一致性,并通过重新验证过程同步卷状态表.

2.3 断连操作

断连是手持设备在移动环境中经常碰到的现象.一旦代理的失效通知得不到客户响应,它就认为客户已经处于断连状态.这时,代理修改自己为该客户保存的卷状态表,然后立即向服务器返回确认信息.为了避免重复检测客户的连接状态,代理还维护了一个断连客户表,对于表中的客户,代理不再尝试发送通知,从而减少了服务器写操作的延迟.

一旦客户重新取得连接,它要首先验证本地缓存是否还有效.来自客户的验证请求说明客户已经重新取得连接,因此,代理可以把它从断连客户列表中删除.客户的重新验证过程描述如下:

对于每个卷,执行下列验证过程:

(1) 和代理比较卷的版本号.

(2) 如果版本号一致,则本卷中的所有对象都是有效的.验证下一个卷.

(3) 如果版本号不一致,则对卷中的每个对象执行如下验证过程:

  (a) 和代理比较对象的上次修改时间.

  (b) 如果修改时间一致,则对象有效.验证下一个对象.

  (c) 否则,该对象在断连期间被修改过,把它从缓存中删除.验证下一个对象.

当重新验证过程完成以后,只有有效的对象还保存在卷中,因此,我们可以重新建立卷上的回调.由于上述验证过程是在客户使用本地缓存之前完成的,因此,我们的算法可以保证数据的强一致性.算法中所采用的“懒惰”方式,即删除脏数据而不是立刻请求新数据,不仅可以加速验证过程,降低通信开销,还有利于同步客户和代理所维护的卷状态表.

2.4 动态卷调整

影响卷回调验证算法性能的一个重要因素是卷的划分.如果每个卷中包含的对象太少,则大粒度验证的优点得不到充分发挥;如果卷中包含的对象太多,则卷验证失败后又会造成过多的对象验证消息.

根据文献[4]的统计结果,Web 文档的修改符合一种冷/热模式,即经常被修改的文档只是所有文档的一小部分,剩下的大部分文档具有比较长的稳定期.因此,对单个文档而言,最近被修改过的文档将来被修改的概率比较大;长时间没有改动的文档将来被修改的概率比较小.基于这个观察,我们在基本卷回调算法的基础上进行了改进,使得卷划分可以动态地进行调整.

卷的调整由拆分和合并两个操作组成.当卷 V 的大小 Size(V)超过上限 S_{max} 时,我们根据对象的最近一次修改时间把它拆成两个新的卷 V_1 和 V_2.拆分过程如下,其中卷中对象以链表方式组织并按照修改时间的升序排列.

(1) 根据卷 V 的属性创建两个新卷 V_1 和 V_2,并加到卷列表中.

(2) 把 V 中的前 S_{min} 个和后 S_{min} 个对象分别移动到 V_1 和 V_2 中.

(3) 获取卷 V 中的第 1 个和最后一个对象的上次修改时间,并求出它们的平均值 T_m.

(4) 对于卷中的每个对象,

(a) 若对象的上次修改时间小于 T_m,则把它移到 V_1 中;

(b) 否则,把它移到 V_2 中.

(5) 从卷列表中删除空卷 V.

上述拆分过程的时间复杂度为 $O(1)$,空间复杂度为 $O(L)$,L 为对象的个数.它满足下面两个条件:

· $S_{min} \le \text{Size}(V_i) \le S_{max}, i=1,2$;

· 从 V_1 和 V_2 中分别任意取出一个对象 O_1 和 O_2,有 $\text{Date}(O_1) \le \text{Date}(O_2)$,Date($O$)是对象 O 的上次修改时间.

综上所述,在客户端缓存中,所有来自同一个 Web 服务器的卷按照其中对象的修改时间升序排列为 $V_1, V_2, \ldots, V_n$.因此,合并算法很简单,当卷 V 的大小 Size(V)小于下限 S_{min} 时,它将与来自同一服务器的相邻的一个卷合并.如果合并后的卷大小又超过上限,则再次应用拆分算法.如果找不到另一个来自同一个服务器的卷,则不进行合并.合并过程的时空复杂度与拆分过程相同.

由于卷的拆分和合并是随着对象的添加和删除进行的,在客户-代理-服务器结构中,动态的拆分和合并过程会导致代理维护的卷信息表和客户维护的卷信息表不一致.例如,代理收到一个新的对象并把它所在的卷拆成两个,然后通知客户,但客户刚好在这时断连,在允许客户断连的情况下无法保证缓存操作和通知的原子性,所以我们允许出现上述不一致的情形.客户在重新验证缓存时会察觉到上述不一致性.在这种情况下,客户对所有来自该服务器的卷直接进行对象粒度验证,从而重新建立卷状态表.

3 算法性能测试

我们在 HTTP/1.1 协议的基础上实现了 ATTL、回调、卷回调和动态卷调整的卷回调这 4 种缓存验证算法.为了比较算法的性能,我们以真实访问记录为原始数据进行了模拟试验.试验的过程和结果如下.

3.1 试验环境

试验环境是在 Linux 平台上建立的,客户和服务器及代理之间采用局域网络连接.由于是模拟试验,我们选择消息数和消息字节数作为度量指标.

原始数据来自 http://ita.ee.lbl.gov.它记录了自 1994 年 11 月到 1995 年 5 月,Boston 大学计算机系实验室中 37 台工作站上用户访问 WWW 的信息[5].在此基础上,我们使用[3,6]中的方法模拟生成了服务器对文档的修改记录.首先随机选择了 10%的文件作为经常修改的文件,让它们每天被修改的概率为 0.25;剩下 90%的文件每天被修改的概率为 0.01.由于原始数据来自固定网络上的浏览器,还需要生成断连记录,生成过程保证每个时间长度超过 30 分钟的浏览期都会出现一次断连.

模拟器由服务器、代理、客户和时间协调器组成.其中,客户进程运行在若干客户机上,通过协调客户访问的时序,我们用了 3 个客户模拟大多数的原始记录;其他 3 部分进程运行在服务器上,其中时间协调器定时发出时间同步消息,协调客户和服务器之间读写操作的时序.

3.2 试验结果

试验结果记录了使用各种验证算法的 Web 访问过程中产生的消息数和消息字节数.我们着重比较无线网

络段的数据传输.

验证消息代表了网络回路,是缓存验证算法的一个主要性能指标.在某些情况下,消息的发送还需要建立新的网络连接,这是导致系统延迟的重要因素.表 2 按月比较了 4 种算法所产生的消息数,其中第 1 栏中列出的是试验过程的总消息数,包括验证消息和请求、响应等,第 2 栏中单独列出了验证消息数.对于 ATTL 算法,验证消息是指存活期过期以后,客户向代理服务器发送验证请求和响应消息;对于回调算法,它是指断连后重新验证所产生的消息.由于每个月的原始数据有比较大的差异,我们着重比较它们之间的相对值.从表 2 可以看出,ATTL 算法产生的验证消息非常少,基本上少于总消息数的 1%.这主要是因为,在模拟试验中,缓存不能跨作业期使用,在持续时间短的作业期内,缓存过期的可能性很小.传统回调算法会造成大量的重新验证消息,而卷回调算法把验证消息削减到它的 21%左右(从 9%~38%).在此基础上,动态卷调整可以进一步提高卷回调算法的性能,减少验证消息数约 22%.数据分布显示这种改进的效果变化幅度比较大,这可能与算法中的参数选择有关.卷回调和 ATTL 算法产生的验证消息总数都小于总消息数的 5%.带动态卷调整的卷回调算法在消息总数上非常接近 ATTL 算法,它产生的消息总数仅比 ATTL 算法多 4%~12%.

Table 2 Performance evaluation (message number)
表 2 性能评估(消息数)

	ATTL		Callback①		Volume callback②		Volume resize③	
	Total messages④	Validation messages⑤	Total messages	Validation messages	Total messages	Validation messages	Total messages	Validation messages
94-11	249	2	405	153	265	18	265	18
94-12	655	2	1 015	282	678	24	683	29
95-01	1 724	9	2 695	832	1 920	205	1 844	129
95-02	6 321	76	10 483	4 232	7 072	821	7 008	757
95-03	4 109	25	6 637	2 549	5 044	956	4 632	544
95-04	4 918	3	8 434	3 519	5 377	462	5 364	449
95-05	1 473	10	2 547	1 009	1 628	163	1 607	142

①回调,②卷回调,③卷调整,④总消息数,⑤验证消息.

表 3 比较了消息字节数,包括请求、响应和验证消息.消息字节数代表了 Web 访问过程的整个网络开销.表 3 中算法的差异小于表 2,这是因为网络开销的大部分是文档传输造成的,因此,缓存验证算法所造成的影响比较小.表 3 的结果和表 2 相符合,在此不再重复.

Table 3 Performance evaluation (message size)
表 3 性能评估(消息大小)

	ATTL	Callback①	Volume callback②	Volume resize③
94-11	27 725	27 808	27 734	27 734
94-12	11 726	13 372	11 739	11 742
95-01	26 944	28 505	27 044	27 006
95-02	79 210	81 353	79 647	79 615
95-03	45 679	46 575	45 967	45 881
95-04	118 370	120 130	118 609	118 595
95-05	14 639	16 665	14 825	14 814

①回调,②卷回调,③卷调整.

4 相关工作

我们的工作基于两方面的研究:一方面是 WWW 中强弱一致性缓存验证算法的比较研究;另一方面是移动环境数据访问中无线网络通信优化的研究.

Liu 和 Cao[3]详细分析了几种常用的缓存验证算法的原理和在 WWW 环境中的效率,研究了在 WWW 中维持缓存数据强一致性的可能性.试验结果表明,基于回调的强一致性验证算法在性能上接近弱一致性的 ATTL 算法.他们尤其指出,与回调算法相比,ATTL 算法节省的通信开销完全是由返回脏数据作为代价而得到的.Yin 等人以契约(lease)算法为基础对卷回调算法进行了改进[6],通过把短时间的卷契约和长时间的对象契约相结合,他们提出了几种算法以降低服务器的写操作延迟,从而增强了系统的可扩充性.和他们的工作相比,我们的工作从移动计算的角度出发,侧重于客户端断连后的验证过程.

缓存验证算法是移动计算系统中优化无线通信的一个重要研究方向.Mummert 等人[7]把大粒度回调算法

应用到 Coda 分布式文件系统中,显著地降低了用户的读操作延迟.Barbara 和 Imielinski[8]提出了一种基于分发的移动环境中缓存验证算法,它利用广播频道来定时播送失效报告(invalidation report),即数据修改信息.客户无须向服务器发送检测消息,只需监听广播频道即可.

5 结 论

数据一致性和效率是广域网络环境下缓存验证算法的两个主要性能指标,移动环境的约束为这个问题引入了新的挑战.传统的 WWW 缓存验证算法以牺牲数据一致性为代价得到了很高的效率.研究表明,增大缓存验证粒度是消除基于回调的缓存验证算法由于客户断连所造成的重新验证瓶颈的有效途径.本文提出了一种改进的基于卷的回调算法,使得移动环境中的缓存验证过程在保持数据的强一致性的前提下接近通用的弱一致性算法的高性能,成功地在数据一致性和性能之间取得了平衡.该算法不仅可以运用于移动环境,也同样可以运用于基于 Internet 的广域网环境.

References:

[1] Satyanarayanan, M. Fundamental challenges in mobile computing. In: Proceedings of the 15th ACM Symposium on Principles of Distributed Computing. Philadelphia: ACM Press, 1996. 1~7.

[2] Floyd, R., Housel, B., Tait, C. Mobile Web access using eNetwork Web express. IEEE Personal Communication Magazine, 1998, 4(5):47~52.

[3] Cao, P., Liu, C. Maintaining strong cache consistency in the World Wide Web. IEEE Transactions on Computers, 1998, 47(4):445~457.

[4] Gwertzman, J., Seltzer, M. World Wide Web cache consistency. In: Proceedings of the USENIX Annual Technical Conference. San Diego: USENIX Association, 1996. 141~152.

[5] Cunha, C., Bestavros, A., Crovella, M. Characteristics of WWW traces. Technical Report, TR-95-010, Department of Computer Science, Boston University, 1995.

[6] Yin, J., Alvisi, L., Dahlin, M. Volume leases for consistency in large-scale systems. IEEE Transactions on Knowledge and Data Engineering, 1999,11(4):563~576.

[7] Mummert, L., Satyanarayanan, M. Large granularity cache coherence for intermittent connectivity. In: Proceedings of the USENIX Summer 1994 Technical Conference. Boston: USENIX Association, 1994. 279~289.

[8] Barbara, S., Imielinski, T. Sleepers and workaholics: caching strategies in mobile environments. In: Proceedings of the 1994 ACM Conference on SIGMOD. Minneapolis: ACM Press, 1994. 1~12.

An Efficient Cache Validation Algorithm for Mobile Web Browsing*

ZHOU Huan[1,2], LI Jing[1], FENG Yu-lin[1,2]

[1](*Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*);

[2](*Technology Center of Software Engineering, Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*)

E-mail: {quan,lij}@otcaix.iscas.ac.cn; feng@ios.ac.cn

http://www.ios.ac.cn

Abstract: In this paper, client-proxy-server based volume callback algorithms are introduced to address two key issues that face the traditional algorithms: scalability and client disconnection. Proxies extend scalability by alleviating loading on servers. Volume based callback accelerates validation process and eliminates revalidation bottleneck when reconnected. Simulation results show that the improved volume callback algorithms save validation messages by 86% compared with the traditional algorithms. And its overall performance is close to ATTL (adaptive time to live) algorithm.

Key words: cache validation algorithm; callback; volume; mobile computing; disconnection

* Received July 26, 2000; accepted August 7, 2000

Supported by the National Natural Science Foundation of China under Grant No.69833030; the National Grand Fundamental Research 973 Program of China under Grant No.G1998030400