基于语义缓存的粒度自适应一致性维护策略①

周红静[1]，杨金民[2]

[1](湖南商学院 计算机与电子工程学院，长沙 410205)
[2](湖南大学 信息科学与工程学院，长沙 410082)

摘　要：对象/关系映射常使用缓存来提升处理性能，缓存中数据与服务器数据的一致性维护是影响系统可靠性的关键问题。提出由中间层发起一致性维护的策略。该策略结合数据存储粒度、数据更新频率及更新数据量大小等因素，分别采用 TTL 及按需请求方式来进行一致性维护，保持缓存数据与服务器数据一致。实验结果表明，该策略能有效减少网络数据传输的开销，降低网络负载，并保证数据的有效性。

关键词：对象关系映射；语义缓存；一致性维护；粒度

Granularity Adaptive Consistency Maintenance Scheme Based on Semantic Caching

ZHOU Hong-Jing[1], YANG Jin-Min[2]

[1](School of Computer and Electronic Engineering, Hunan University of Commerce, Changsha 410205, China)
[2](School of Information Science and Engineering, Hunan University, Changsha 410082, China)

Abstract: Cache is used to improve performance in object-relational mapping. Data consistency maintenance between cache and server is the key factor of system reliability. In this paper, a granularity adaptive consistency maintenance scheme that is initiated by middle layer is proposed. This scheme considers several data attributes including storage granularity, updating frequency and size. After combining the number of cache items, it uses TTL and on-demand request respectively. Experimental results show that the proposed policy can effectively reduce network overhead and load. Data efficiency can also be guaranteed.

Key words: object relational mapping; semantic cache; consistency maintenance; granularity

当前软件开发中经常使用面向对象技术实现业务逻辑的设计，使用关系数据库存储数据。对象关系映射（object relational mapping, ORM）[1]能够完成对象模型和关系模型的转换，使应用程序以面向对象的方式来访问关系数据。缓存有利于提高大部分数据操作的性能[2]，但也带来了副本数据一致性维护的复杂性和较大的开销。随着数据库服务器中的数据更新，副本数据也必须修改对原对象的映射，并保持与服务器数据库中的数据一致。一致性包括强一致性和弱一致性。强一致性策略主要有广播失效机制和服务器主动失效机制[3]。大多数基于 Web 的应用对于内容一致性没有严格的要求，一般采用弱一致性维护策略，数据更新可由服务器方或客户机方发起，分为客户端轮询、定期作废、回调传值、回调作废、按需请求及定期更新[4]等方式。这些方法一般采用全属性更新，数据更新频率较高时，会带来很大的开销。文献[5]指出 ORM 中不同的缓存结构、粒度对系统有不同的影响。本文将讨论在 ORM 中间层应用语义缓存，依据应用的数据更新频率和数据粒度来设计一致性维护策略，以减少一致性维护带来的开销对缓存性能的影响。

1　语义缓存技术

语义缓存是将数据访问结果和相关的语义描述信息缓存,利用语义为以后的数据查询提供解答的一种缓存手段[6]。语义缓存作为一种重要的缓存技术，近几年在多方面得到了深入的研究。文献[7]针对语义缓

① 基金项目:湖南省自然科学基金(10JJ3041);湖南省教育厅资助科研项目(11C0744);长沙市科技局科技计划(K10ZD062-13)
收稿时间:2011-10-09;收到修改稿时间:2011-11-17

存的半结构化查询提出了一个近似保守算法，并分析了分布式查询处理的适用性。但该方法只适用的场景有限，缺乏通用性。文献[8]提出在客户端通过采用协作的方式共享他们本地语义缓存扩展一般的语义缓存机制，来提高服务器的吞吐量，但需要增加客户端之间管理的开销。文献[9]针对移动网络环境，将更新的粒度细化到更新的属性来实现语义缓存一致性维护，以便减小通信开销。但该方法缺乏对缓存项数据量的优化。

语义缓存中的缓存项粒度并不统一，它是满足一定条件的记录集，而且语义缓存支持关系数据库查询，具有更灵活、更高效等特点。另一方面，在对象系统中类之间的继承、关联等关系是对象之间属性、行为相关程度的体现。因而在 ORM 系统中对查询采用语义缓存方法，正好可以通过语义描述信息来权衡数据查询与缓存项之间的语义相似程度，选择保留与访问相似性较大的缓存项来达到重用缓存数据的目的。下面先给出语义缓存的定义，然后以此为基础给出相应的缓存一致性维护策略。

定义 1. 已知一给定的关系数据库 $R=\{R_i|1\leqslant i\leqslant n\}$，其中 R_i 表示关系，语义缓存定义为 $SC=\{<CA_i, CR_i, CP_i, CC_i>|1\leqslant i\leqslant m\}$，CAi 表示第 i 个缓存项包含的属性集，$CR_i$ 表示第 i 个缓存项对应的关系，对$\forall i, j, 1\leqslant i, j\leqslant m$, $CR_i\neq CR_j$。CP_i 表示第 i 个缓存项对应的条件集，CC_i 表示第 i 个缓存项所包含的记录集，$CC_i=\Pi CA_i(\sigma CP_i(CR_i))$。

定义 2. 查询 Q 表示为四元组<QA, QR, QP, QC>，其中 QA 表示要查询的属性集，QR 表示待查询的关系，QP 表示查询满足的条件集，QC 表示记录集，$QC=\Pi QA(\sigma QP(QR))$。

2 一致性维护策略

2.1 ORM 中间层一致性维护的特殊性

本文所讨论的基于语义缓存的一致性维护具有的特殊性在于：

(1) 在逻辑结构上，它是应用在中间层的缓存一致性维护，是客户端缓存管理的一种延伸。

(2) 由于采用语义缓存，缓存中保留的是查询描述及其满足一定查询条件的结果集，当服务器中的数据发生变化时，如果缓存中拥有这样的数据，那么就很可能会出现数据值变化后缓存数据和其语义之间发生冲突，一般主要包括数据错误冲突和数据不完整冲突。

(3) 缓存中一些缓存块对应的关系之间也可能是关联的，当某些缓存块数据出现上述冲突时，与其关联的缓存块数据的完整性也就有可能被破坏，这样也会对用户的使用造成影响。

2.2 关键问题处理

(1) 缓存空间大小给定时，尽量减少缓存项数目及数据冗余，降低一致性管理及维护开销。随着每一次新的数据访问的到来，中间层缓存中的缓存项逐步增多时，不同的语义缓存项之间就可能会出现某些数据和语义上的重叠，因此，在适当的条件下对语义缓存项进行合并。这样可以增大后续数据访问的完全命中概率，使语义缓存维护中的重复工作大为减少；另外，缓存项的合并，有利于查询裁剪的实现，降低查询匹配的开销。

(2) 区别对待缓存中改变频率高低不等的数据项，减轻一致性维护的网络传输数据量。采用语义缓存方式存储的记录集数据，基本上可以分为：恒定不变的数据，只发生增量的数据，偶尔改变的数据和经常改变的数据。设置计数器 Counter 表示访问属性项的次数，当 Counter 值达到或超过某一上限值，认为该域的数据是经常改变的，否则便是偶尔改变的数据，并以此作为数据垂直分片的依据。通过数据垂直分片，将不同的属性项在进行一致性维护的时候进行分类，调整缓存数据粒度。随着数据访问的周期性变化，垂直划分的数据也会相应地产生变化，由此数据更新的粒度也将随着数据访热点的变化而改变。

(3) 设定数据服务器总是保持数据的最新版本。对于在一致性维护过程中产生的服务器数据与中间层缓存的数据冲突问题，由于服务器中并没有保存缓存项数据的相关语义，因此只需要维护中间层缓存数据和服务器数据的一致性，而无需处理缓存数据和其语义描述之间的冲突。对经常改变的数据，在一个 TTL 周期就会完成从服务器端的数据更新。而偶尔改变的数据则可以逐步延长更新数据时间。

2.3 粒度自适应一致性维护策略

由于在本策略中将会使用到语义缓存项的合并，下面先给出有关定义。

定义 3. 给定语义缓存项 $SC_1=<CA_1, CR_1, CP_1, CC_1>$和 $SC_2=<CA_2, CR_2, CP_2, CC_2>$，如果 $CR_1=CR_2$，则称 SC_1 和 SC_2 是可合并的语义缓存项。

根据合并扩展的数据内容的不同，将语义缓存的合并分为条件合并、属性合并和混合合并，同时通过

上述的三种合并方式完成有效的查询裁剪。

定义 4. 给定语义缓存项 SC_1=<CA_1, CR_1, CP_1, CC_1>和 SC_2=<CA_2, CR_2, CP_2, CC_2>，如果 CA_1=CA_2，CR_1=CR_2，$CP_1 \neq CP_2$，生成的新缓存项 SC_3=<CA_1, CR_1, $CP_1 \cup CP_2$, $CC_1 \cup CC_2$>则称为条件合并。

定义 5 给定语义缓存项 SC_1=<CA_1, CR_1, CP_1, CC_1>和 SC_2=<CA_2, CR_2, CP_2, CC_2>，如果 $CA_1 \neq CA_2$，CR_1=CR_2，CP_1=CP_2，生成的新缓存项 SC_3=<$CA_1 \cup CA_2$,CR_1,CP_1,$CC_1 \cup CC_2$>则称为属性合并。

定义 6 给定语义缓存项 SC_1=<CA_1, CR_1, CP_1, CC_1>和 SC_2=<CA_2, CR_2, CP_2, CC_2>，如果 $CA_1 \neq CA_2$，CR_1=CR_2，$CP_1 \neq CP_2$，生成的新缓存项 SC_3=<$CA_1 \cup CA_2$, CR_1, $CP_1 \cup CP_2$, $CC_1 \cup CC_2$>则称为混合合并。

中间层发起一致性维护的实现过程如下：

(1) 取单位时间预热数据,设预期数据更新频度为 k。

(2) 将中间层语义缓存中来自于同一关系的缓存项 SC_i 和 SC_j(1≤i≤j≤n)合并得到 SC_i，并简化该缓存项的语义描述。

(3) 获取缓存中各 SC_i 块中对应属性项的 Counter 值，将 SC_i 重新进行垂直分割，得到除主关键字外互不相交的两部分 SC_{i1} 和 SC_{i2}，其中 SC_{i1} 为基本不变的数据，SC_{i2} 为经常更新的数据(其中 SC_{i2} 中数据对应字段的 Counter > 1/k)。

(4) 设定 SC_{i1}(1≤ i ≤n)的 TTL 时间。

(5) 当查询请求提交给中间层时，如果命中缓存项 SC_i，且命中数据属于 SC_{i2}，则提交缓存中的 SC_i 的语义描述信息至服务器，并依据时间戳与服务器端数据进行比较，若该版本旧于服务器端数据则从服务器返回这部分数据，并对缓存中的 SC_{i1} 部分数据进行一致性维护，否则修改中间层和服务器端的时间戳，并将中间层修改后的数据传回服务器端，保持数据一致；若数据命中 SC_{i1} 部分，则直接从中间层获取访问数据，并等待 TTL 时间到达转(6)；若缓存未命中则转(7)。

(6) 当 SC_{i1}(1≤i≤n)指定的 TTL 时间到达时，则与数据服务器进行连接，进行两端数据的一致性维护。

(7) 将查询请求提交服务器，从服务器取回数据并使用替换策略将该部分数据存入缓存。

3 性能测试与分析

实验硬件环境包括一台数据库服务器（安装 SQL Server 2000），一台应用服务器（安装有 Hibernate 和 Java 程序）和三台客户机。应用系统中建立了 12 个类，这 12 个类分属 5 个不同的族，其中最大的类有 1000 个不同对象，最小类有 200 个对象，每个族中的对象类最多具有 3 个重复值属性，非重复属性值最多不超过 5 个。测试了当 SC_{i1}/SC_i 的比例大小发生变化时，粒度自适应维护策略的一致性维护平均响应时间和平均数据量，并和传统的轮询、按需请求的一致性维护策略以及复制控制法[10]进行比较。

3.1 平均响应时间与平均传输数据量测试

在本实验中，缓存命中时是可以直接从中间层缓存获取数据的，然而，缓存未命中时，在该实验中无论使用到哪一种一致性维护策略，都相当于从服务器获取并更新缓存数据，其时间和开销是一致的。因此在测试中所得到的平均响应时间和传输数据量的大小均指的是缓存命中时的情况。

取 SC_{i1}/SC_i 的平均比值大小分别为 10%~50%，测试数据由 1000 个随机查询序列构成，后 900 个查询的执行结果作为比较数据，SC_{i1} 的 TTL 时间设置为 24 小时，轮询的时间取为 10 分钟，最高更新频率数据是每 10 分钟更新一次，缓存比例分别取为 15%和 30%，实验结果如图 1 到 4 所示。

从图 1 和图 3 中可以看出，在基本不变的数据量占缓存总量的比值不断增大时，粒度自适应的维护策略显然比中间层的轮询、按需请求策略及复制控制法具有更快的维护响应时间。从图 2 和图 4 中可以看出，粒度自适应的维护策略也比其他策略维护的数据量要更小。原因在于轮询每次在一定时间段中需将所有的缓存项的时间戳与服务器中对应数据的时间戳进行比较，而按需请求只是在缓存命中时将该缓存项的更新请求发送到服务器，虽然粒度自适应策略有部分数据是按需请求，有部分数据是根据其 TTL 时间来进行更新，但当 TTL 时间较大时，这部分数据的更新时间和数据量要远少于经常更新的数据部分所需的对应开销，而且随着不太变化的数据量在缓存中总比重的增加，这种开销的减少愈发明显；复制控制法则是传输整个元组集到客户端，然后逐条维护的机制，相比较而言，粒度自适应的策略在更新的数据量上也更有优势。另外，数据更新的频率实际上也是一个重要的参照指标，如果数据更新频率越高，整个缓存的一致性维护的响应时间和实际传输的数据量都会加大。

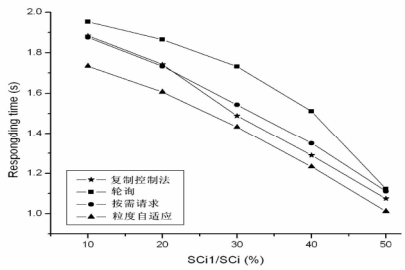

图 1 维护的平均响应时间对比测试 (cache=15%)

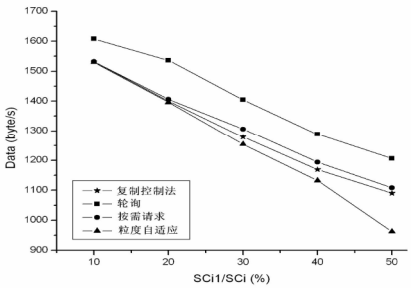

图 2 维护平均传输数据量对比测试 (cache=15%)

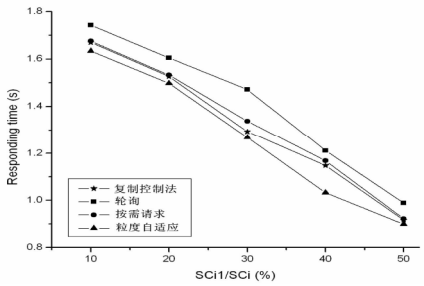

图 3 维护平均响应时间对比测试 (cache=30%)

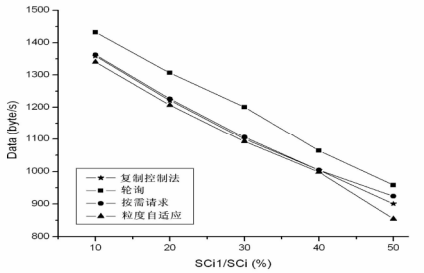

图 4 维护平均传输数据量对比测试 (cache=30%)

4 结语

缓存提升了系统的性能和可靠性，但由此所带来的成本是必须维护缓存副本数据与服务器数据的一致性。文中提出了由中间层发起的一致性维护策略。该策略结合语义缓存特点，首先通过合并语义缓存项尽量减少缓存项的数目和数据冗余，并根据数据更新频率的大小，将基本不变的数据部分采用 TTL 策略思想，而经常改变的数据则采用按需请求方式来分别进行一致性维护，缓存中数据的更新的粒度会随着需求的变化而变化，同时在数据库数据更新的前提下，保持缓存数据与其一致。该策略不需对缓存的数据项进行全属性数据传输，能有效减少网络数据传输的开销，降低网络负载，并保证数据的有效性。

参考文献

1 Keller W. Object/relational access layers–a roadmap, missing links and more patterns. Proc. of EuroPlop'98. MA: Addison Wesley. 1998. 1–25.

2 Zyl PV, Kourie DG, Coetzee L, et al. The influence of optimisations on the performance of an object relational mapping tool. Proc. of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists. New York: ACM, 2009. 150–159.

3 Cao P, Liu CJ. Maintaining strong cache consistency in the world-wide web. IEEE Trans. on Computers, 1998,47(4): 445–457.

4 蔡建宇.面向海量数据库的中间层语义缓存技术研究.长沙:国防科学技术大学,2005.

5 Keith M, Stafford R. Exposing the ORM cache. ACM Queue. 2008,6(3):38–47.

6 Dar S, Franklin MJ, Jonsson BT. Semantic data caching and replacement. Proc. of the 22nd VLDB Conference. San Francisco, CA: Morgan Kaufmann Publishers Inc, 1996. 330–341.

7 Novikov B, Pigul A, Yarygina A. A performance analysis of semantic caching for distributed semi-structured query processing. Proc. of the 14th East European Conference on Advances in Databases and Information Systems. 2011. Berlin, Heidelberg: Springer-Verlag, 2011: 421–434.

8 Vancea A, Stiller B. Answering queries using cooperative semantic caching. Proc. of the 3rd International Conference on Autonomous Infrastructure, Management and Security, 2009. Berlin, Germany: Springer-Verlag, 2009: 203–206.

9 李东,袁应化,叶友,等.基于属性更新的语义缓存一致性维护算法.华南理工大学学报(自然科学版),2009,37(5):139–144.

10 陈眠,喻丹丹,涂国庆.分布式数据库系统中数据一致性维护方法研究.国防科技大学学报,2002,24(3):76–80.