语义缓存的一致性维护策略研究

李 东[1] 刘振宇[1] 杨小鹏[1] 叶 友[2]

(华南理工大学软件学院 广州 510006)[1] (华南理工大学计算机科学与工程学院 广州 510640)[2]

摘 要 为了解决在移动环境下将更新操作转化成删除和添加操作的传统一致性维护算法所增加的不必要的数据通信流量和数据存取,现将语义裁剪的思想融入一致性维护算法,将更新粒度细化至被更新的属性,并将更新的语义区域裁剪到最小。理论分析表明通过对更新范围的裁剪,有效减少了数据存取和数据通信的开销。仿真实验证明了该策略的可靠性和高效性,特别是当更新数据流大的时候尤其明显。

关键词 语义缓存,一致性维护,语义裁剪

中图法分类号 TP301 **文献标识符** A

Study of Semantic Caching Coherency Scheme

LI Dong[1] LIU Zhen-yu[1] YANG Xiao-peng[1] YE You[2]

(Dept. of Software, South China University of Technology, Guangzhou 510006, China)[1]

(Dept. of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China)[2]

Abstract Semantic Caching update operation was usually replaced by delete operation and insert operations in mobile environments. In order to reduce the addition of unnecessary data communications flow and data storage brought by traditional coherency strategy, this paper proposed a new reliable strategy for update operations via brought the idea of semantic trimming into update operations. The granularity of update operations reached the exact size of the attributes, and the semantic regional needed to update was cut to the minimum. Theoretical analysis shows that by trimming the scope of the update, it effectively reduces data access and data communications costs. The reliability and high efficiency of this new semantic caching coherency control scheme were examined and analyzed through a simulation study. Especially when updating the data flow was large, the superiority was more obvious.

Keywords Semantic caching, Coherency control scheme, Semantic trimming

1 引言

无线移动数据访问越来越成为人们获取各种动态变化的数据信息的一个重要手段,但是移动计算环境具有低带宽、网络断接频繁、设备移动性和移动设备资源有限等特征[1,2]也成为移动数据存取的瓶颈。目前,主要采用在客户端缓存经常访问数据的方法来减少网络负荷和提高响应时间。但是,传统的缓存技术并不能很好地解决移动计算环境通信的固有劣势[3]。语义缓存技术是一种基于结果集和相应描述的缓存技术。相对页缓存和元组缓存技术,它能更好地节约网络开销,节省缓存容量,支持网络断接等。这使得语义缓存技术在移动计算环境下有非常广阔的应用前景。目前在语义缓存技术的研究上已经取得了一定的理论成果,文献[3]中 Ren Qun 给出了语义缓存的形式化表述,研究了用户查询处理和缓存管理机制;并指出语义缓存对缓存一致性策略提出了新的要求。文献[4]中对如何从语义缓存导出当前查询结果的问题出发,给出了查询从缓存导出的充分条件,并在定义查询与缓存之间的精确匹配,包含匹配和相交匹配关系的基础上,给出缓存与查询的包含和相交的判断条件及其相应算法。文献[5]把谓词分成范围型和约束型,比较单个谓词的逻辑差,不同类型的谓词在逻辑运算担任不用的角色,提出范围相交定理,在此定理之上,利用范围的相交和约束型的满足与否来确定剩余查询。文献[6,7]给出了语义缓存不一致的定义,并给出了将更新语句转化为删除和添加操作的算法及其建立在此基础之上的一致性维护策略。

综上所述,虽然语义缓存相对于传统缓存有明显的优势,但是由于语义缓存模型不仅在客户端缓存了服务器数据的副本,而且还缓存了数据的语义描述,这使得语义缓存一致性问题更加复杂。目前有关语义缓存一致性的研究大多处于理论研究阶段,研究并设计出可靠且实用的语义缓存一致性维护策略是十分重要的。

本文的主要贡献包括:第一,形式化定义了语义缓存一致性维护相关的概念;第二,通过将更新语句的语义与本地缓存语义取交集,精确锁定更新区域,将数据存取减少到最小;第三,通过对现有移动通信缓存一致性维护模型的研究,提出了一种新的有状态按需请求的强一致性维护策略,将更新操作

到稿日期:2008-01-30 本文受广东省自然科学基金资助项目(480B6040550)资助。

李 东(1970-),男,博士,副教授,主要从事数据库、XML 和移动计算方面的研究,E-mail: cslidong@scut.edu.cn;刘振宇(1984-),硕士,主要从事数据库和移动计算方面的研究;杨小鹏(1982-),硕士,主要从事数据库和移动计算方面的研究;叶 友(1980-),男,硕士,主要从事数据库和移动计算方面的研究。

所消耗的通信负载减少到最小;第四,通过仿真实验及实验结果分析证明了该更新算法的易实施性和其相应的一致性维护策略的高效性和可靠性。

2 语义缓存一致性问题分析

2.1 传统语义缓存一致性维护算法

传统的语义缓存一致性维护算法[9]是基于将 Update 操作转化为 Delete 操作和 Insert 操作的算法,文献[9]中虽然提出了应该采用先更新数据,再维护语义的方法,但是其更新操作的算法仍然是基于传统算法的思想。基于将 Update 操作转化为 Delete 操作和 Insert 操作的语义缓存一致性维护算法[9]有以下几个缺点:

1. 客户端更新的语义区域完全取决于服务器端更新语句所决定的语义区域,冗余了客户端不必要更新的区域及相应数据,因此增加了客户端的数据处理量及加重了通信负载,并延迟了更新响应时间。

2. 由于采取将 Update 操作转化为 Delete 操作和 Insert 操作,客户端必须重新从服务器端索取该更新语句的语义所代表的存储在服务器端的所有数据(如①更新语句中的 age<=30 在服务器数据库中代表的所有数据)。这样处理极大地加重了通信带宽不必要的负载,并增加了更新操作时间。

3. 由于从执行插入操作之前要比较每个元组与语义的相关性,设元组的数量为 m,语义段的数量为 n,缓存语义条件谓词平均属性数量 k,则比较的时间复杂度为 $O(n*m*k)$,极大地增加了客户端 CPU 的开销和更新操作时间。

4. 由于传统算法中对于每一个 update 语句都需要连接服务器端的数据库进行操作,并且更新的粒度为关系表中的元组,因此更新操作负担了不必要的时间延迟及大量不必要的数据存取。

2.2 基于语义裁剪的语义缓存一致性维护算法

针对上一节总结的传统缓存一致性维护算法所固有的缺点,本文提出了基于语义裁减的语义缓存一致性算法。

定义 1(更新属性) 更新语句(Update 语句)所涉及的关系 R 中被更新的属性集合称为更新属性集合,用 U_A 表示,则缓存查询谓词属性集合用 S_{PA} 表示。

定义 2(关键更新属性) 令更新属性集合与语义缓存片的查询谓词属性集合的交集称为该更新语句关于该语义缓存片的关键更新属性集合 US_A,即 $US_A=U_A\cap S_{PA}$,其相应的谓词为关键更新属性谓词,即 US_{AP}。

定义 3(附属更新条件谓词) 更新条件谓词中与更新属性集合无关的谓词子集称为附属更新条件谓词,记为 Up'。

定义 4(关键更新属性谓词满足) 若关键更新属性非空,关键更新属性谓词 US_{AP} 满足缓存的查询谓词 S_P 成立,称为关键更新属性谓词满足,即 $US_{AP}\rightarrow S_P$ 成立。

定义 5(关键更新属性谓词冲突) 若关键更新属性非空,关键更新属性谓词 US_{AP} 满足缓存的查询谓词 S_P 不成立,称为关键更新属性谓词冲突,即 $US_{AP}\rightarrow S_P$ 不成立。

根据以上定义可以得到基于语义裁剪的更新算法:

(1)对于 Update 语句的更新操作,遍历语义缓存片,对每个语义缓存片作如下操作:

1. 若关键更新属性为空,取更新条件谓词与缓存查询谓词的交集,若该交集为空,则跳过该语义段,继续遍历;若该谓词交集非空,则基于该谓词交集构建更新语句(Update R set U_{AP} where $Up\wedge S_P$)对客户端数据缓存操作,继续遍历剩余语义段。

2. 若关键更新属性非空,关键更新属性谓词冲突,取更新条件谓词和缓存查询谓词交集,若该谓词交集为空,则跳过该语义段,继续遍历;若该谓词交集非空,则基于该谓词交集构建删除语句(Delete form R where $Up\wedge S_P$)对客户端数据缓存进行操作,继续遍历剩余语义段。

3. 关键更新属性非空,关键更新属性谓词满足,取关键更新属性谓词,附属更新条件谓词和缓存查询谓词三者的交集,若该谓词交集为空,则跳过该语义段,继续遍历;若该谓词交集非空,则进行以下两个步骤后再继续遍历剩余语义段:

第一步 取更新条件谓词和缓存谓词的交集,若该交集为空,则跳到步骤二,若该交集非空,则基于该交集构建新的更新语句(Update R set U_{AP} where $Up\wedge S_P$);

第二步 基于更新属性谓词和更新附属条件谓词和缓存查询谓词三者的交集构建新的查询语句(Select * from R where $U_{AP}\wedge S_P\wedge Up'$)。从服务器取数据插入到客户端,并在客户端数据缓存中剔除重复的元组。

(2)对于 Delete 语句的更新操作,在客户端直接利用该 Delete 语句删除缓存数据。

(3)对于 Insert 语句的更新操作。当在服务器端发生了插入操作,首先遍历所缓存的语义段,由于各语义段是不相关的,当该要插入的元组符合任意一个语义段的语义时,元组立即被插入数据缓存。

定理 1 按照基于语义裁剪的更新算法完成操作后,语义缓存 SC 的一致性是可以满足的。

证明:对于 delete 和 insert 操作的更新规则在文献[9]中已经给出了详细的证明过程,此处不再重复。以下给出对于 update 操作的更新规则。

假设 t 时刻客户端缓存一致性满足,$t1$ 时刻服务器端发起 update 操作。(1)若该 update 操作中关键更新属性为空,则服务器没有产生满足客户端缓存语义 S_P 的新元组,因此不存在数据不完整冲突。若更新条件谓词与缓存查询谓词的交集 $Up\wedge S_P$ 为空,则说明不存在数据错误冲突,此时缓存一致性满足;若 $Up\wedge S_P$ 非空,则说明满足此谓词交集的数据集将造成数据错误冲突,根据具更新规则在 $t2$ 进行(Update R set U_{AP} where $Up\wedge S_P$)操作即可消除数据错误冲突,此时缓存一致性满足。

(2)若该 update 操作中关键更新属性非空且关键更新属性谓词冲突,则服务器没有产生满足客户端缓存语义 S_P 的新元组,因此不存在数据不完整冲突。由于更新操作使得满足 U_p 的数据集合不再满足 S_P,因此,客户端缓存中满足 $Up\wedge S_P$ 的数据集合必然造成数据错误冲突,根据具更新规则在 $t2$ 进行(Delete form R where $Up\wedge S_P$)操作即可消除数据错误冲突,此时缓存一致性满足。

(3)若该 update 操作中关键更新属性非空且关键更新属性谓词满足,若 $U_P\wedge S_P$ 非空,则说明满足此谓词交集的数据集将造成数据错误冲突,根据具更新规则在 $t2$ 进行(Update R set U_{AP} where $Up\wedge S_P$)操作即可消除数据错误冲突,若 $U_{AP}\wedge S_P\wedge Up'$ 非空,则说明满足此谓词交集的数据集将造成

数据不完整冲突，因此(Select * from R where $U_{AP} \wedge S_P \wedge Up'$)从服务器取数据可以消除数据不完整冲突，但由于 $Up \wedge S_P$ 和 $U_{AP} \wedge S_P \wedge Up'$ 所代表的数据集合存在，因此在客户端应消除重复的数据，此时，缓存一致性满足。于是，根据该更新算法进行更新维护可以使得缓存一致性满足。证毕。

3 语义缓存模型

3.1 客户端模型

客户端模型的存储核心基于 xml 文件形式，这样处理有以下几点优势：1)客户端只需要提供一个 xml 文件解析器，适合于手机移动应用；2)数据存储以语义为标示分段存储，减少数据访问开销；3)统一标准格式，方便维护和应用扩展；4)很好地解决了移动用户关机后需要重新缓存的问题。

客户端主要有 3 类 xml 表，分别是元数据表、语义表和数据表。其中，元数据表的功能是充当语义缓存的数据字典，存储服务器数据库中相应的基表的详细信息；语义表的功能是为语义裁剪和缓存一致性维护提供语义历史记录，存储经过语义裁剪后的最终查询语句，将查询语句中的重要元素拆分为节点；数据表的功能是基于语义段的划分来缓存数据，是数据访问范围精确到语义范围。

3.2 服务器端模型

服务器端模型的主要功能是确定与服务器数据库的更新/插入/删除操作相关的客户端，并对以上操作进行优化。

定义 6(客户端索引 CI[9]) CI 形如$\langle R,Q,C\rangle$，其中，R 是服务器的关系或视图，Q 是查询，C 是对关系或视图 R 进行了查询的客户端，即 $C=\sigma_Q(R)$。

定义 7(更新队列 U_i[9]) U_i 是由于某客户端 i 有关的插入、删除或更新操作组成的队列。

4 语义缓存一致性维护策略

为了使语义缓存技术更好地适用于基于 GSM，GPRS 和 UMTS 等协议的商用移动通信环境，并通过研究无线环境下传统的缓存一致性策略[9-13]，本文提出了一种新的有状态按需请求的强一致性维护策略。图 1 给出了该策略的通信模型。

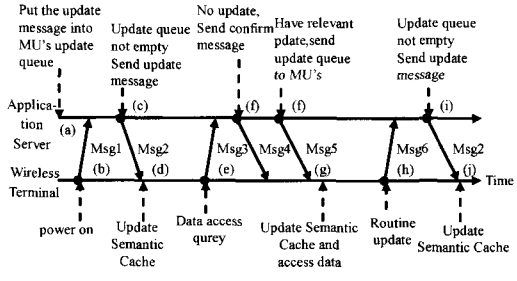

图 1 客户端-服务器通信模型

服务器端：

1. 应用服务器若收到来自数据库服务器的更新消息，则扫描缓存的客户端语义信息，将与之相关的更新操作信息插入到每个客户的更新队列中，见图中(a)；2. 若服务器收到客户端的开机消息或者轮询消息，见图中(b,h)，则检查该用户的更新队列中是否有更新操作信息存在，如果存在，则将更新队列中的信息(msg2)发送到该客户端，如果更新队列为空，则发送确认数据有效的消息(msg4)到客户端；3. 如果服务器接收到客户发送的查询消息，则扫描缓存的客户更新队列，如果有与探测查询语义相关的更新信息，则发送该更新信息(msg5)到客户端，否则发送确认数据有效的消息(msg4)到客户端。

客户端：

1. 客户端开机时即向服务器端发送启动消息，若接收到确认数据有效消息(msg4)，则置位缓存有效消息，若收到更新队列信息(msg2)，则对本地缓存进行更新；2. 若用户通过客户端访问数据，则客户端发送数据请求消息(msg3)到服务器端，之后若客户端收到确认数据有效消息(msg4)，则立即从本地访问缓存的数据，并与剩余查询获得数据合并返回给客户。若客户端收到更新队列信息(msg2)，则将缓存更新后再将查询的数据返回给客户。若客户端时钟到达轮询时段，则发送轮询消息(msg6)到服务器，若接收到确认数据有效消息，则置位缓存有效消息，若收到更新队列，则对本地缓存进行更新。

从以上介绍的客户端与服务器端的一致性维护策略算法以及通信消息可以看出，首先相对于基于 IR(Invalidate Report)广播机制[8-12]的一致性维护策略，本文提出的有状态按需请求的强一致性维护策略可以更好地解决移动环境下客户端无规律断接的情况，减少了监听所需要消耗的客户端能耗，并有效地解决了由于断接时间超出更新时间标签有效范围需要将客户端缓存全部清空的问题。其次，由于服务器为每个客户维护了更新队列，并记录了每个客户的缓存语义，因此服务器可以精确锁定用户请求数据的精确范围，减少了对数据的访问量并缩短了查询的响应时间。再次，相对于传统的按需请求的强一致性维护策略，本文提出的一致性维护策略由于增加了动态周期间隔的轮询机制，并且在访问数据时只探测服务器端是否有关于指定语义所代表的数据的更新，因此有效地缩短了客户访问数据的响应时间。

5 模拟实验与性能分析

5.1 模拟仿真模型

我们设计了模拟仿真实验来检验本文提出的语义缓存一致性维护策略的性能，特别是本文提出的基于语义裁剪的客户端更新算法相对于传统的客户端更新算法的优势，由于一致性维护非常复杂，要考虑很多因素，为了简化问题，数据更新以由服务器所提交的更新为准作为触发。

(1)语义缓存模型

应用服务器，客户端三者通过 LAN 连接组成。模拟实验系统框架如图 2 所示。

语义缓存是基于 xml 表的，当客户端启动时，语义信息和元数据信息会载入内存中以 hash 表的形式存在，每个客户端只需要一个 xml 解析器就可以完成对数据和语义信息的操作。客户端查询处理模块进行查询裁剪，并将剩余查询语义及查询结果分别写入语义表和相应的数据表中；一致性维护模块完成对更新操作的解析及最终完成对缓存的更新操作。查询产生器用来在客户端随机产生用户查询，以模拟客户对数据的请求；应用服务器端主要完成维护每个客户的更新队列及监视更新日志；数据服务器由数据库系统及更新生成器组成，模拟对服务器端数据的更新维护操作。

(2)模拟实验环境

模拟实验环境基于 J2SE 平台，数据库服务器为 SQL SERVER 2000。客户端操作系统和服务器端操作系统均为 windows2003。实验机器 CPU 为 Pentium 4 2.8G 内存 512M，硬盘 80G。数据库采用某公司 CRM 系统的历史数据

进行测试，测试的数据库中包括 6 个关系，包括了字符型、整数型、实数型、Date 型 4 类属性（其中有 3 个 integer 属性，1 个 float 属性，3 个 varchar 属性；integer 属性为 4Bytes，float 属性为 8Bytes，varchar 属性为 10Bytes），有 200000 多条元组。数据分布在规定的范围之类，方便测试。语义缓存和数据在客户端均以 xml 文件形式表示。

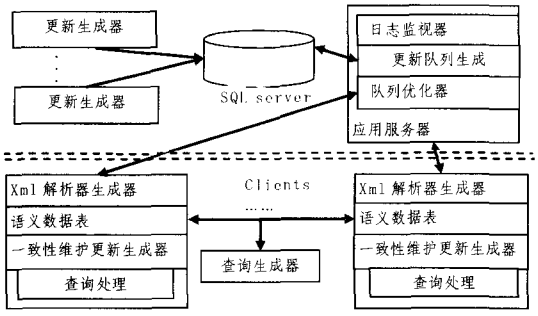

图 2　系统框架

5.2　实验结果分析

(1)客户端更新维护性能分析

本模拟实验选择两项主要指标，即缓存更新时间消耗和网络通信量，对比分析了基于语义裁剪的客户端更新算法对减少更新操作的无用功和网络通信量方面的优势。

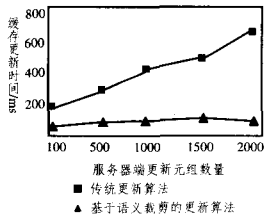

图 3　无新插入元组更新操作响应时间分析图

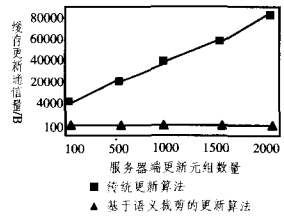

图 4　无新插入元组的更新操作网络通信量分析图

如图 3,4 所示，当进行 update 操作无额外的元组插入到客户端缓存中，即第 2 节所述基于语义裁剪的更新规则中的前两种情况时，由于基于语义裁剪的更新算法避免了不必要的数据通信和数据解析过程，更新操作完全在本地完成，且更新粒度精确到属性，因此通信量仅为更新语句所占的字节数，响应时间并不会随着数据量的增加而显著增加；而传统算法必须从服务器取数据，取回的每一个元组都必须和缓存语义比较，因此响应时间和网络通信量都随着服务器端更新操作所涉及的元组数量增加而线性增加。

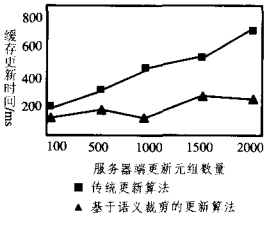

图 5　有新插入元组的更新操作响应时间分析图

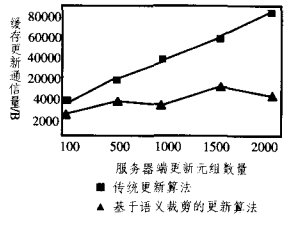

图 6　有新插入元组的更新操作网络通信量分析图

如图 5,6 所示，当进行 update 操作有额外的元组插入到客户端缓存中，即第 2 节所述基于语义裁剪的更新规则中的第三种情况时，由于客户端向服务器端请求数据的查询语义是更新属性谓词和更新附属条件谓词和缓存查询谓词三者的交集，相对于传统算法中仅依靠更新属性谓词所构成的语义向服务器端请求数据的方式，大量地减少了无用的数据存取和通信。图 5,6 中分别明显地说明了本算法在减少网络负载指标和响应时间指标上的优势。

综合以上分析，本文提出的基于语义裁剪的语义缓存一致性维护策略在响应时间和网络负载两个指标上都明显优于传统语义缓存一致性维护策略，所以本文提出的基于语义裁剪的语义缓存一致性维护策略是可靠而高效的。

结束语　本文详细介绍了目前语义缓存一致性维护研究中采取的传统的客户端更新算法的操作流程，并清楚地归纳出该传统算法在数据通信量、本地数据存取、响应时间 3 方面存在的不足。将更新语义与缓存语义进行比较和裁剪，提出了基于语义裁剪的维护语义缓存一致性的更新规则及约束条件，并进行了形式化描述和证明；在此基础上，通过对移动环境下传统缓存一致性维护策略及语义缓存一致性维护策略，提出了适用基于 GSM，GPRS 和 UMTS 等协议的商用移动通信环境的按需请求的强一致性语义缓存维护策略。本文详细描述了为实现语义缓存一致性的两个重要算法：客户端一致性更新维护算法和按需请求的强一致性算法。最后，通过模拟仿真实验证明应用本文提出的策略不仅可以维护客户端语义缓存数据和服务器端数据的一致性，并且在通信带宽占用，更新响应时间和本地数据存取 3 个方面相对传统的客户端更新算法都有明显的优势。

我们今后还将针对由客户端发起的数据更新机制设计语义缓存一致性维护策略，并结合本文提出的语义缓存一致性维护策略研究更适用于现代商用移动环境的语义缓存一致性维护策略。

参 考 文 献

[1] Forman G, Zahorjan J. The challenge of mobile computing. IEEE Computer, 1994, 27(6): 38-47

[2] Wu K L, Yu P S, Chen M S. Energy-efficient Caching for Wireless Mobile Computing // Proc. 12th International Conference on Data Engineering. Feb. 1996

[3] Ren Qun, Dunham M H, Kumar V. Semantic Caching and Query Processing. IEEE Trans on Knowledge and Data Engineering, 2003, 15(1): 192-210

[4] 吴婷婷，周兴铭. 基于语义缓存的移动查询导出. 计算机学报，2002, 25 (10): 1104-1110

[5] 李东，杨小鹏，罗鹏飞. 基于谓词分类的语义缓存查询裁剪. 华南理工大学学报：自然科学版，2008(1)

[6] 万海，郝小卫，章陶，等. 语义缓存一致性维护策略的设计与实现. NDBC2004. 计算机研究与发展，2004, 41(Suppl Oct): 28-34

[7] 郝小卫，章陶，李磊. 移动计算环境下语义缓存一致性维护策略的优化技术. 计算机科学，2005, 32(7 增刊 B)

[8] Cao G H. A scalable low-latency cache invalidation strategy for mobile environments[J]. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(5): 1251-1265

[9] Kahola A, Kuhrana S. A strategy to manage cache consistency in A disconnected distributed environment[J]. IEEE Trans on Parallel and Distributed Systems, 2001, 12(7): 686-700

[10] Jingj , Elmagarmid , Helala , et al. Bit - sequences : an adaptive cache invalidation method in mobile client/server environment [J]. The ACM · Baltzer Journal on Special Topics in Mobile Networks and Applications, 1997, 2(2): 115-127

[11] Cai J T , OOIB. An evaluation of cache invalidation strategies in wireless environments[J]. IEEE Trans on Parallel and Distributed Systems, 2001, 12(8): 789-897

[12] Dass W Z, Arm K. SACCS: scalable asynchronous cache consistency scheme for mobile environments[A] // Proc. of 23rd International Conference on Distributed Computing Systems Workshops[C]. Providence, RI, USA, 2003: 797-802