温和一致性代理复制机制 MCARM *)

吴 劲 卢显良 任立勇 侯孟书
(电子科技大学计算机科学与工程学院 成都 610054)

摘 要 提出了一种新的复制机制:温和一致性代理复制机制(MCARM)。MCARM 采用了主节点的复制管理器与辅助节点的 MSS-Agent 协调工作的架构,吸取严格一致性协议和弱一致性协议的优势,又避开其局限性和复杂性,更好地适应移动计算环境的要求,并能与缓存失效策略 CISBMA 协同工作,较好满足了移动应用的需求。

关键词 数据复制,移动计算环境,主节点,辅助节点

Moderate Consistency Agent Replication Mechanism

WU Jing LU Xian-Liang REN Li-Yong HOU Meng-Shu
(College of Computer Science & Engineering, UESTC of China, Chengdu 610054)

Abstract A new mechanism of moderate consistency agent replication mechanism (MCARM) is presented. It adopts a harmonious framework of replication manager in master node and MSS-Agent in secondary node, which has advantage of the tight consistency protocol and loose consistency protocol respectively while avoiding their insufficiency and complexity at the same time so as to adapt well to the mobile computing environments. MCARM can work with CISBMA in harmony to meet mobile application's demand.

Keywords Data replication, Mobile computing environments, Master node, Secondary node

1 引言

数据复制的关键问题是如何维护多个复制结点上数据状态的一致性。现有的复制协议可分为严格一致性(Tight Consistency)协议和弱一致性(Loose Consistency)协议。严格一致性协议,如 ROWA 协议[1]、主拷贝协议[2]和表决协议[3]等,它们要求在任何时刻所有数据库的复制都是一致的,这样能够确保数据库复制的严格一致性,但这样也会产生可用性与访问性能等方面的问题。在移动计算环境下,为了事务有较短的响应时间,通常我们需要在每个无线网络单元内设置一个复制服务器,这样复制节点的规模较大,因此,传统的严格一致性协议不能满足移动计算应用的需要。

从目前的研究情况来看,大多数移动计算环境下的复制系统采用的都是弱一致性协议,允许在系统中存在暂时的不一致性,但这种不一致通常是保持在一定的界限内,而且总是能够趋于最终的一致状态,如 Lazy-Master 与 Lazy-Group 协议[4]。弱一致性复制协议具有更好的可用性和访问性能,但是也存在一些技术障碍,如解决多复制之间融合时的冲突检测与消解问题,保证系统的收敛性等等,而且,即使是弱一致性应用也可能在必要的时候要求严格的一致性。

因此,我们希望能有一个方案,既可吸取严格一致性协议和弱一致性协议的优势,又能避开其局限性和复杂性,更好地适应移动计算环境的要求。基于文[5]的 DMABMA 体系,我们提出了一种"温和一致性代理复制机制(Moderate Consistency Agent Replication Mechanism,简称:MCARM)",其目标是提高移动计算环境中数据的可用性、可靠性以及访问性能,使整个系统可以支持大量移动主机的并发访问。

2 MCARM 的总体设计

2.1 MCARM 的基本思想

我们的研究工作基于图 1 所示的系统模型,需要复制的数据服务器位于固定网络之中,而在每个 MSS(Mobile Support Station)上都有该数据服务器的一个复本,可以为本单元内的移动主机提供服务。MCARM 实际上是处于 DMABMA 的固定网络部分,可以利用传统的分布式复制策略,但是,DMABMA 要支持大量的移动主机,因此,在确定复制策略时,必须考虑移动计算环境的特点,同时要与基于移动代理的缓存失效方案(CISBMA)[6]相互配合工作,以改善系统的整体性能。

MCARM 采用温和一致性复制策略,从最终用户的角度,与弱一致性复制策略类似,每个复制服务器都支持查询与更新操作,并且允许各个复本之间存在暂时的不一致。因此,当移动用户在访问服务器时,只需要访问服务器的一个复本即可,往往是访问本地 MSS 上的复本。

与弱一致性复制策略不同的地方在于更新事务的处理方式。当移动主机向 Agent 提交了更新事务时,Agent 立即向复本的主服务器提交此事务,并将其记入暂态事务日志中,然后立即给移动主机客户返回其标识符和结果,移动客户不必等待服务器将结果传递给其它复制服务器,因此具有较短的响应时间。

MCARM 采用了一种广播更新的同步方式,即主服务器负责广播其更新信息,使各个 MSS 上的复本同步,这种同步方法能保证或接近数据的紧密一致性要求。为了维护不同服务器上并发事务的一致性,我们采用基于时间戳的并发控制

*)本文由电子信息产业发展基金和电子科技大学校青年基金资助,项目编号分别为:[2002]11006 和 YF020803。吴 劲 博士,主要研究方向为计算机网络及分布式数据库技术;卢显良 教授,博士生导师,主要研究方向为计算机网络技术及应用。

方法[7]，按照事务的主服务器的时间戳对所有其它服务器上提交的事务排序，在每次同步过程中按照时间戳的顺序重排事务日志，因此能够保证每个复本的事务执行次序都是一致的，从而保证了各个复制系统的最终一致性。

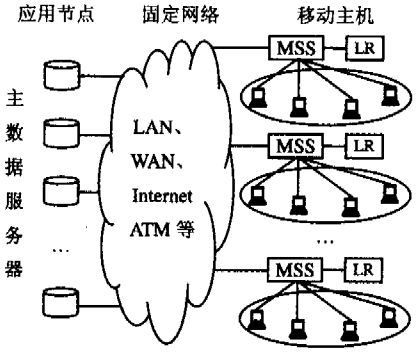

图 1　系统模型

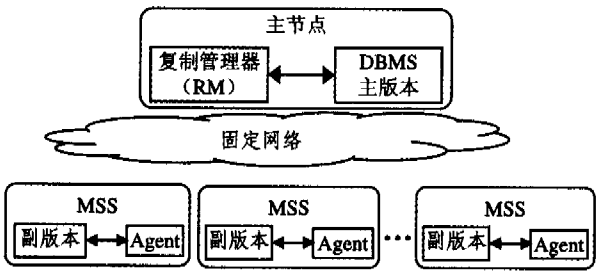

图 2　MCARF 的总体框架

2.2　MCARM 的总体框架

MCARM 的总体框架如图 2 所示，在该模式下数据更新首先提交给主节点（Master Node）上的主版本（Primary Copy），然后通过刷新事务（Refresh Transaction）异步地更新辅助节点（Secondary Node）的副版本（Secondary Copy）。我们使用文[8]的四个参数来说明我们的框架：主权（Ownership）、广播（Propagation）、刷新（Refreshment）和配置（Configuration），当这四个参数设置好，复制方案的框架结构也就出来了。

“主权”参数是文[4]提出来的，如果一个节点拥有“主权”，表明该节点可以更新它的复本。主权节点的复本称为主版本，用大写的 R 表示；而其它版本称为副版本（Secondary Copy），用小写的 r 表示；例如，MCARM 的复本结构为（R，r1，r2，r3，…）。在实际的复制方案中一个节点可以是：主节点（Master Node）、附属节点（Slave Node）和主附节点（Master-Slave Node）。主节点拥有主版本；而附属节点拥有的是副版本；主附节点表明它所拥有的版本既可以是主版本也可以是副版本。

我们按文[2]的概念可以把读写复本的操作分为三种类型：更新（Update）事务、刷新（Refresh）事务和查询。更新事务是对主版本进行更新，对每个更新事务都可以产生一个时戳（Timestamp）；刷新事务是由一系列更新操作构成的事务，其目标是对副版本进行刷新；查询是指对复本的只读操作。

“配置”参数定义了复制方案的节点组织结构，例如：“1Master/nSlave”结构、“mMaster/-nSlave”结构、“Master-slave”结构。MCARM 是采用的“1Master/nSlave”结构。

“广播”参数定义当主版本更新后什么时候向附属节点广播刷新消息。我们关注两类的广播方式：立即（Immediate）方式和延迟（Deferred）方式。立即方式是指当更新事务更新主版本时，一旦主版本被更新就立即发送消息 m 给各个副版本，不用等待整个更新事务的提交。而延迟方式是指当更新事务更新主版本时，一旦主版本被更新就把消息写入 M，当整个更新事务提交后，才把 M 发送给各个副版本。

“刷新”参数定义了刷新事务如何与广播策略协调工作，刷新有两种触发方式：立即（Immediate）方式和等待（Wait）方式，下面我们看一下广播和刷新相互配合而形成的三种策略。“延迟-立即”策略指当采用延迟方式广播更新时，附属节点一收到消息 M，就立即更新副版本。“立即-立即”策略指当采用立即方式广播更新时，附属节点一收到消息 m，就立即更新副版本，不用等待主节点的更新事务的提交。“立即-等待”策略指当采用立即方式广播更新时，附属节点一收到消息 m，就立即更新副版本，但整个刷新事务的提交必须等待主节点的更新事务的提交后才能执行。我们的框架里不涉及消息的交换机制，我们假设复制系统建立在可靠的 FIFO 广播协议[9]基础上。

3　MCARM 的功能设计

3.1　MCARM 的主节点

MCARM 的主节点的结构如图 3 所示。与数据复制相关的模块包括复制管理器（Replication Manager，简称：RM）和网络接口模块（Network Interface Module，简称：NIM）。复制管理器的主要功能模块包括：事务代理（Transaction Agent）、变化捕获器（Change Capturer）、分发器（Distributor）；另外还包括两个日志文件：事务日志（Transaction Log）和变化日志（Change Log）。

为了独立于 DBMS，网络接口模块负责 RM 与各个辅助节点之间的通信，NIM 包含一个网络连接表，其表记录格式为：（Slave_id，MH_id，路由信息）。“事务代理”接收来自 MSS 的请求，把它转发给 DBMS，同时把请求内容记录在“事务日志”中，事务日志格式为：（Slave_id，MH_id，Status，Transaction），Status 字段的值可以是：待执行（Wait）、已提交（Committed）、放弃（Abort）；DBMS 的响应也由“事务代理”接收，然后转发给相应的 MSS，同时根据响应情况修改事务日志的 Status 字段。“变化捕获器”的主要功能是提取事务日志或 DBMS 的逻辑日志中的更新操作，并写入用于刷新辅助节点的副版本的“变化日志”，变化日志里的事务都会被赋予主节点的时戳（Timestamp）。“分发器”的作用可以分成两部分：一是当某个 MSS 第一次或者系统崩溃了需要复本时，负责把复本传送给相应的 MSS；二是负责把更新信息广泛传播（propagation）给相应的 MSS。

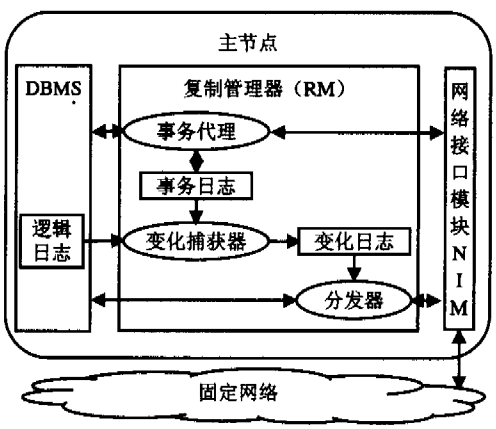

图 3　MCARF 的主节点结构图

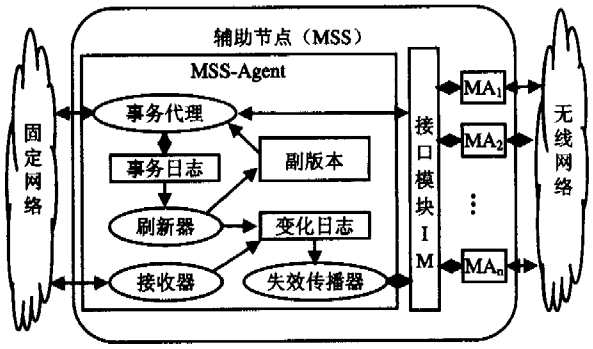

图 4　MCARF 的辅助节点结构图

3.2　MCARM 的分发功能

数据分发是数据复制的重要功能，许多辅助功能模块都是为数据分发能正常工作而存在的，所以在这一小节，我们讨论一下和数据分发相关的概念。分发由不同的节点承担会导致产生不同的分发模型，从而影响复制的组织形式和效率：如果由源节点承担分发任务，传播内容表现为由源节点主动地发送给目标节点，这就是推式(push)模型；而如果由目标节点承担分发任务，传播内容表现为由目标节点向源节点申请而来，这就是拉式(pull)模型。两者相比，推式模型的优势在于效率较高，而拉式模型在于容易调度，要考虑到源节点可以主动减少传播内容方面的因素，这些差别会因参与节点数量的增多而表现得更加明显，大多数复制方案同时支持这两种模型。

MCARM 的分发器对这两种模式都支持，一是当某个 MSS 第一次或者系统崩溃了需要复本时，采用拉式模型，由 MSS 向主节点分发器发出申请，分发器才把复本传送给相应的 MSS；二是一旦辅助节点(MSS)上的复本建立成功，以后分发器采用推式模型，把更新信息广泛传播(propagation)给相应的 MSS。

复制初始化、崩溃恢复时，MCARM 的分发器采用的是完整拷贝(Full Copy)方式，将需要的复制对象传输给相应节点，这种方式下的传输量等于复制对象的大小。一旦辅助节点(MSS)上的复本建立成功，以后 MCARM 的分发器采用增量修改(Incremental Update)方式进行辅助节点的刷新，这种方式下的传输数据是复制对象的全部变化(修改、插入、删除)序列，传输量主要由事务数、单个事务修改数据项平均数目和平均数据项大小决定，并随复制类型的不同有所变化。

增量修改的最大优点就在于它是唯一符合单复本可串行性(one-copy serializability)原则的更新方式。单复本可串行性是指应同时满足单复本等价性和事务并发调度的可串行性的更新方式，因此能提供详细的控制信息，可以适用于任何类型的复制。它的缺点是需要一些特定机制的支持，需要的资源也比较多。

还有一种称为净变化(Net Change)的更新方式。传播的数据是始末两个时刻复制对象的净变化值，这种方式下传输量为所有发生变化的数据项数目与平均数据项大小的乘积。显然，净变化方式的传输量最小，而且这种方式有多种实现方法，这是它的优点；但是它的缺点也很明显，它不符合单复本可串行性的要求，中间过程的操作信息全部丢失，不能提供足够的控制信息。由于它不以分布式事务为基础，不能用于同步复制，而且提供的控制信息相对简单，在对等式复制时也会遇到问题，因此在我们的 MCARM 分发器没有采用这种分发方式。

3.3　MCARM 的辅助节点

MCARM 的辅助节点的结构如图 4 所示。与数据复制相关的功能部件包括 MSS-Agent、接口模块(Interface Module，简称：IM)，以及各移动主机 MH(Mobile Host)在 MSS 上的移动代理 MA(Mobile Agent)。MSS-Agent 的主要功能部件包括：事务代理(Transaction Agent)、接收器(Receiver)、刷新器(Refresher)、失效传播器(Invalidation Propagator)；另外还包括两个日志文件：事务日志和变化日志。接口模块 IM 负责 MSS-Agent 与各个 MA 之间的通信。

“事务代理”接收来自各个 MA 的请求，如果是查询请求，先查询本地副版本，如果能满足要求则直接返回给相应的 MA；如果不能满足要求或是更新请求，就把它转发给主节点上的复制管理器，同时把请求内容记录在“事务日志”中，事务日志格式为：(MH_id，Status，Transaction)，Status 字段的值可以是：待执行(Wait)、已提交(Committed)、放弃(Abort)；DBMS 的响应也由“事务代理”接收，然后转发给相应的 MA，同时根据相应情况修改事务日志的 Status 字段。

“接收器”的主要功能是从网络接口模块接收更新消息，并写入用于刷新辅助节点的副版本的“变化日志”。“刷新器”的功能是根据相应的刷新策略，利用变化日志，对副版本进行数据刷新。“失效传播器”是一个附属部件，与数据复制没有直接关系，只是利用数据复制产生的信息与我们提出的缓存失效方案 CISBMA[6] 配合工作。失效传播器提取变化日志的信息，向各个 MA 广播缓存失效消息。

4　性能分析

4.1　相关的研究工作与比较

4.1.1　复制模式比较　数据复制模式可以分成主从模式和对等模式[4]。主从复制模式(Master/Slave)，也称单向(unidirectional)复制或主版本(master copy)复制。只有主节点才能对主复本进行更新，其他节点向拥有该数据的主节点订阅(subscribe)数据。在该模式下复制总是按同一个方向进行，其它目标节点上的复本是只读的，如果想对它进行修改，必须先修改主节点上的主复本，然后再同步到目标节点的复本上，这样就能从根本上预防复制冲突的发生。主从式复制在流程和组织上都相对简单，而且不同复本的数据模式可以有差别。

对等复制模式(peer to peer)，也称双向(bidirectional)复制或多版本(multi-copy)复制或随处修改(Update Anywhere)复制。复制可同时在两个方向进行，每个节点上的复本都是可读写的，修改其中任何一个都会最终影响全部复本，因此冲突在理论上不可避免，所以必须要有复杂的冲突检测和消解机制[4]来辅助此模式的实现。

目前在移动计算领域的复制技术大多采用的是等对模式，如国防科技大学的李霖等提出的面向移动领域的 TTR，它是一个由服务器级复制、空中复制和客户级缓存组成的三级复制体系结构，它的服务器级的复制采用的就是对等复制模式。另外，大部分的研究工作把移动主机也纳入数据复制节点，虽然具有较大的灵活性，但是实现复杂，特别是在数据一致性管理上的开销很大。通过我们的研究、分析和比较发现，移动计算环境普遍采用图 1 的网络基础模型，如果利用该体系结构，可以采用主从复制模式，大大简化复制的组织和流程，而且采用中间件技术还可以解决异构数据复制的问题。

4.1.2 变化捕获方式比较 变化捕获是数据库复制的基础，它直接决定了数据库复制的刷新方式，对其它环节的影响也非常大，下面对几种变化捕获方式进行分析和比较。

快照(Snapshot)是数据库中存储对象在某一时刻的即时映像。通过为复制对象定义一个快照或采用类似方法，可以将它的当前映像作为更新复本的内容。基于快照法是最简单的变化捕获方法，可以在任何数据库甚至是其它结构化和半结构化的数据源上实现。它不需依赖特别的机制，不占用额外的系统资源，管理和操作也非常容易，而且在复制初始化和崩溃恢复时是必须的。但由于无法区分复制对象中哪些具体项发生改变，因此效率很低，一般不用于正常情况下的数据刷新。

基于触发器法[11]是在源数据库为复制对象创建相应的触发器，当对复制对象进行修改、插入和删除等命令时，触发器被唤醒，将变化传播到目标数据库，通常这需要 RPC、分布式 SQL 的协助。基于触发器法克服了基于快照法的主要缺点，极大提高了效率，如果辅以其它机制，可以用于同步复制和对等式复制。但是触发器捕获法占用的系统资源比较多，对较复杂的复制任务需要非常复杂的配置和实施，管理极不方便。

数据库日志作为维护数据完整性和数据库恢复的重要工具，其中已经包含了全部成功提交的数据库操作记录信息。基于日志法就是通过分析数据库日志来捕获复制对象的变化序列，利用日志，复制对象的变化很容易在其它节点再现，这不但能提高效率和保证数据的完整性，还能在对等式复制时提供详细的控制信息。但是基于日志法也存在一些缺点，首先，一些数据库系统不公开其日志的格式，除非厂家提供相应的日志分析工具或接口，否则要开发一个基于日志的变化捕获程序比较困难；其次，尽管都是利用数据库日志获取变化，但不同数据库系统在具体细节上还是存在很大差异，这会给异构数据库复制带来新的问题；最后，很多情况下 DBA 对数据库日志的管理已经很繁重、复杂，而基于日志法无疑会更加加重这种负担。

基于 API 法可以在应用程序和数据库之间引入中间件，由它提供一系列 API，这些中间件在完成应用程序对数据库修改的同时，也把复制对象的变化序列记录下来，从而达到捕获的目的。基于 API 方法能够实现基于日志法的大多数优点，而且会给异构数据库复制带来便利，也不再增添 DBA 的负担。但是基于 API 法也存在缺点，对那些不经过 API 的操作，如在控制台(console)命令行方式下键入 Update、Insert、Delete 语句，它们所引起的数据变化是 API 无法捕获到的。

影子(shadow)表法通过比较对象表 T(初始化时复制映像)和影子表 S(当前映像)的内容来获取净变化信息。影子表法是另一种通用的捕获方法，能在任何数据库上实现，应用程序可以方便地在多种平台间移植，因此很适合解决异构数据库复制。影子表法的代价只有一倍的存储空间和不高的管理成本，由于得到的是净变化值，传输效率还能进一步提高。但影子表法的缺点也很明显。首先，它不符合单复本可串行性的要求，中间过程的操作信息全部丢失，不能提供足够的控制信息；其次，在判断操作时间先后时，不可避免地会产生 FCLD(First Change Last Detect)现象，这样在对等式复制时就会出现许多问题。最后，每次捕获变化都需扫描整个 T 表和 S 表，捕获效率很低，随着节点数目的增多会成为一个严重的性能瓶颈。

通过对多种变化捕获方法的分析和比较，我们发现基于 API 法和基于日志法比较适合 MCARM 的要求，在我们的方案里主要是采用中间件和日志法相结合的方法，扬长避短，既给异构数据复制带来便利，又能符合单复本可串行性的要求，保证了缓存失效策略 CISBMA 的顺利实施。

4.2 性能测试

我们的模拟实验由主节点和辅助节点构成，数据库根据 Wisconsin Benchmark 规范设置，包含三个表：onektup，tenktup1 和 tenktup2；三个表的结构相同，每个表包含 13 个整数列、3 个 52 字节的字符列；其中 onektup 包含 1000 个元组，tenktup1 和 tenktup2 各包含 10000 各元组。

我们测试的目标是复制节点的副版本的刷新效果，为此我们按照文[4,12]的方式，定义了刷新度(Refreshness Degree)rd，其定义如下：

$$rd=1-\frac{n(R)-n(r)}{n(R)} \tag{1}$$

其中 $n(R)$表示已成功提交给主节点的主版本更新事务的数量，$n(r)$表示已成功提交给辅助节点的副版本刷新事务的数量。$rd\in[0,1]$，如果 rd 接近于 0 表示复本刷新执行效果很差，接近于 1 表示刷新效果极佳。另外我们还定义了一个测试参数 ltr，它表示长事务率，即长事务占整个事务总数的比例，我们按照下面的方式设置了实验环境：

情况 1：$ltr=0$(所有的更新事务都是短事务)，

情况 1：$ltr=30$(30%的更新事务是长事务)，

情况 1：$ltr=60$(60%的更新事务是长事务)，

情况 1：$ltr=100$(所有的更新事务都是长事务)。

下面我们来考察一下广播和刷新相互配合而形成的三种策略：延迟-立即策略、立即-立即策略、立即-等待策略的刷新度与长事务率的关系。

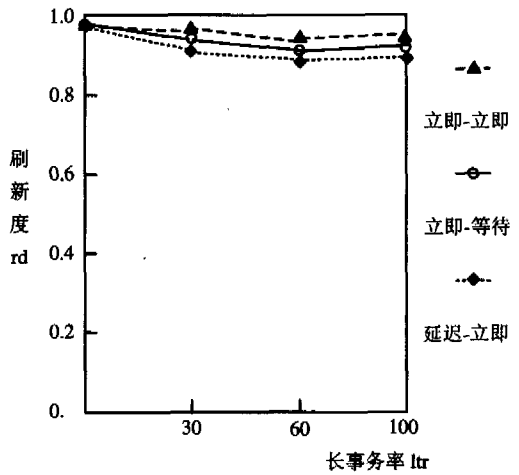

图 5 低更新事务到达率 f-l 图

我们首先考察主节点在较低的更新事务到达率的情况下的刷新度，如图 5 所示。从图中可以看出，当 $ltr=0$ 时，三种策略的刷新度都接近于 1，这是因为，对于短事务而言，当更新事务 T_i 在主节点的主版本 R 上提交后，在 T_{i+1} 提交给 R 之前，刷新事务 T_i 被成功提交给辅助节点副版本 r 的可能性极高。随着长事务率的提高，总体来说刷新度降低；但当 ltr 接近于 1 时，刷新度反而稍稍有些提高，这是应为当更新事务 T_i 在主节点的主版本 R 上提交后，由于事务较长，在 T_{i+1} 提交给 R 之前，刷新事务 T_i 被成功提交给辅助节点副版本 r

(下转第 67 页)

还可以用于互联网络上的 Web service 的 QoS 分析监测等方面。

参 考 文 献

1 Foster I, Kesselman C. The Grid: Blueprint for Future Computing Infrastructure [M]. San Francisco, USA: Morgan Kaufmann Publishers, 1999

2 Segal B. Grid Computing: The European Data Project [A]. In: IEEE Nuclear Science Symposium and Medical Imaging Conference [C], Lyon, 2000. 15～20

3 Siweiorek D P, Swarz R S. The Theory and Practice of Reliable System Design. Digital Press, Digital Equipment Coporation, ISBN 0-932376-13-4, 1983

4 leon J. Fail-safe PVM. PVM Users Griup Meeting, Oak Ridge, TN, May 1994

5 Wittenbrink C M. Survey of Parallel Volume rendering Algorithms. In: Proc. of Int. Conf. On Parallel Distributed Processing Techniques and Applications [C]. 1998. 1329～1336

6 Mahovsky J, Benedicenti L. An Architecture for Java-Based Real-time Distributed Visualization. IEEE Transactions on Visualization and Computer Graphics [J]. 2003, 9(4): 570～579

7 Bethel E W, Shalf J. Cactus and Visapult: An Ultra-High performance Grid-Distributed Visualization Architecture Using Connectionless Protocols. IEEE Computer Graphics and Applications [J]. 2003, 23(2): 51～59

8 Li K, Malony A, Bell R, et al. A Framework for Online Performance Analysis and Visualization of Large-Scale Parallel Applications. International Conference on Parallel Proceeding and Applied Mathematics [C], Czestochowa, Poland, September 2003

9 Brunst H, malony A, Shende S, et al. Online Remote Trace Analysis of Parallel Applications on High-performance Cluster. International Symposium on High-performance Computing [C], October 2003

10 OGSA. http://www.gridforum.org/ogsi-wg/drafts/GS_Spec_draft03_2002-07-17.pdf

11 Foster I, Kesselman C, Nick J, et al. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. January, 2002. http://www.globus.org/research/papers/ogsa.pdf

12 Web Service. http://www.w3.org/2003/ws

13 Soap. phttp://www.w3.org/TR/Soap

(上接第 61 页)

的可能性也稍稍提高。从图中还可以看出延迟-立即方式和立即-等待方式的更新度比较接近，而立即-立即方式是三种中更新度最高的。

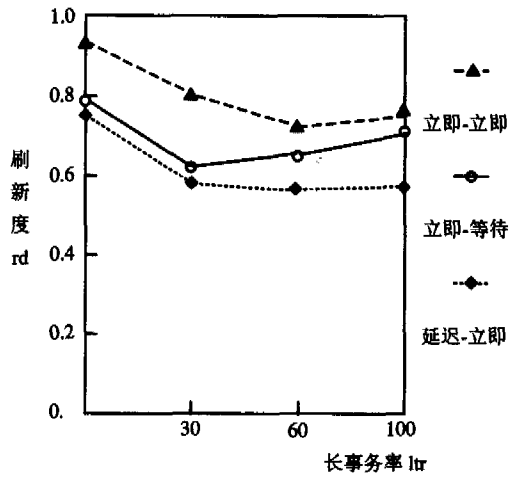

图 6 高更新事务到达率 f-1 图

我们接着考察主节点在高的更新事务到达率的情况下的刷新度，如图 6 所示。从图中可以看出，当 $ltr=0$ 时，三种策略的刷新度都比低更新事务到达率的情况下的刷新度低，这是因为，对于短的频繁到达的短事务而言，当更新事务 T_i 在主节点的主版本 R 上提交后，刷新事务 Ti 被成功提交给辅助节点副版本 r 前，T_{i+1}、T_{i+2}、T_{i+3}…已经提交给 R 的可能性极高。在高事务到达率的情况下，网络相对比较拥塞，刷新度受网络的影响比较大，延迟传播方式比立即传播方式所需的传播时间更长，因此当长事务率较高后，立即-等待比延迟-立即方式的更新度要高，而且差距加大，这同在低到达率情况下的状态差别比较大。

总结 本文提出了一种“温和一致性代理复制机制 MCARM)”，该机制采用了主节点的复制管理器(RM)与辅助节点的 MSS-Agent 协调工作的架构，吸取严格一致性协议和弱一致性协议的优势，又避开其局限性和复杂性，更好地适应移动计算环境的要求，并能与缓存失效策略 CISBMA[6] 协同工作，保证了移动用户的应用需求，具有较强的实际应用价值。

参 考 文 献

1 Agrawal D, Bernstein A J. A nonblocking quorum consensus protocol for replicated data. IEEE Trans. Parallel Distrib. Syst. April, 1991. 171～179

2 Bernstein P A, Goodman N. Concurrency control and recovery in database systems. ACM Comput. Surv, 1981, 13(2): 185～221

3 Gifford D. Weighted voting for replicated data. In: Proc. 7th ACM-SIGOPS Symposium on OS Principles, Pacific Grove, CA, Dec, 1979. 150～159

4 Gray J, Helland P. The dangers of replication and a solution. Proc. ACM SIGMOD Int. Conf. on Magnagement of Data, Montreal, Canada. ACM Press, New York, June 1996, 25(2): 173～182

5 吴劲，卢显良，任立勇，等. 移动计算环境中基于移动代理的数据管理体系结构. 计算机科学，2005，32(5)：76～78

6 吴劲，卢显良，任立勇. 在移动计算环境中基于移动代理的缓存失效方案. 计算机科学，2003，30(4)：82～84

7 Ceri S, Owicki S. On the use of optimistic methods for concurrency control in distributed databases. In: Proceedings 6th Berkeley Workshop on Distributed Data Management and Computer Networks, Berkeley, Calif, 1982. 117～130

8 Pacitti E, Simon E. Update propagation strategies to improve freshnes in lazy master replicated databases. the VLDB Journal, 2000. 305～318

9 Hadzilacos V, Toueg S. A modular approach to fault-tolerant broadcasts and related problems: [Technical Report TR-94-1425]. Dept. of Computer Science, Cornell University, Ithaca, N. Y., 1994

10 Phatak S H, Badrinath B R. Multiversion reconciliation for mobile database. In: Proc the 15th International Conference on Data Engineering, Sydney, Australia, 1999. 582～589

11 Didriksen T, Galindo-Legaria C A, Dahle E. Database de-centralization: A practical approach. In: Proceedings of the 21st VLDB conference, Sep 1995. 654～665

12 Son S H, Zhang F J. Real-time replication control for distributed database systems: Algorithms and their performance. In: Proceedings of the 4th International Conference on Database Systems for Advanced Applications, Apr 1995. 214～221