计算机科学1996 Vol.23 №.3

数据复制和数据一致性

金煜利 楼荣生

（复旦大学计算机系 上海200433）

摘 要 Replication is currently a highlight of distributed database technology. This article will discuss the relationship between data replication and data consistency. We will introduce data replication and its implementation, and will compare with 2PC protocol and try to present an algorithm to meet data consistency.

关键词 Distributed database, Data replication, Mult-replication, Data consistency.

分布式数据库(DDB)对全局关系分片，把各分片分别存放在不同节点，当应用程序对全局关系进行操作时，DDB将全局关系操作转换为分片操作，利用本地处理速度执行分片操作以提高效率。80年代末，数据库厂商推出了具有各种分布式处理能力的产品，然而实践表明，这种分布式2PC协议和远程操作的方案并不完善。数据节点增加后，DBMS会忙于协调各远程节点之间的提交操作，各事务的响应时间因此变长而使用户处于长期等待之中。

目前普遍采用的解决方案是数据复制技术，其主要思想是：作数据复本并把复本存放在本地节点用于数据库操作。对多节点的分布式数据库系统来说，数据复制至少有两点好处，一是增加系统的可用性，即在有些节点不能工作的情况下也能完成某些事务；二是提高操作的效率，有复本的节点可就近读到复本中的数据，减少了通讯量。

1 数据复制及其实现

根据数据的修改过程与复制过程的关系，复制器可分为同步复制器和异步复制器。同步复制(也称紧密一致性)采用2PC协议，原始复本和远程复本的修改保持同步，所有复本的修改在同一工作单元完成，这些修改要么一起成功，要么一起失败，保证了最高程度的数据完整性。当复本数增加，通讯故障会使某一复本不可用，因而使修改操作失败，导致事务失败概率增加。异步复制(或称松散一致性)提供了另外一种机制，首先提交原始节点复本的修改操作，然后把修改传播到其它复本上。如果某一复本不可用，则该修改将被保存起来等待下一次传播；可用复本则快速地执行复制操作，只需几秒钟时间。如果原始复本修改已提交且存在未修改的某复本，即该复本和原始复本不一致，这也会导致数据库操作失败。不一致复本的修改因数据库失败只能由本地的程序完成，而不由复制系统来控制。

1.1 Sybase复制服务器

异构环境下，Sybase复制服务器使用复制数据模型支持松散事务一致性。这一模型中，数据源是一个特定的数据拷贝，一般称为主复本；数据复制的目标是其它的数据拷贝，一般称为从属复本。修改只能在主复本上进行，然后复制给从属复本。主复本节点上的数据修改操作提交后，才传送到从属复本的节点上去。图1中，数据服务器是可执行数据操作和事

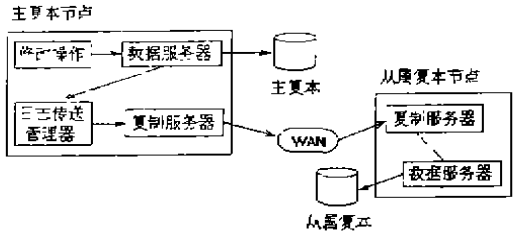


图1 Sybase复制服务器

务处理命令的服务器，它不一定是Sybase或其它关系数据库。日志传送管理器(LTM)读取数据库事务日志，并将需要复制的操作传送给复制服务器。含有主复本数据的数据库和要执行复制存储过程的数据库，必须有LTM。只含有从属复本数据的数据库或没有复制存储过程的数据库不需要LTM。每个节点的复制服务器通过LTM从数据库接收主复本的修

改操作，并将这些修改传播到已声明需要这些数据的节点，由这些节点上的复制服务器修改从属复本。

所有数据的修改必须在主复本节点上执行。从属复本是主复本的只读复本，这样可有效地防止冲突的发生。但使用复制存储过程，客户可以异步请求对主复本的更新。从属复本节点的复制服务器把过程调用发送给主复本节点的复制服务器，并在主复本节点上执行数据修改操作。修改过的数据会被复制到客户应用程序要访问的复本中。表的复制不一定是原始表的全部，可以限于从属复本节点需要的行，通过使用包含一个 where 子句的命令来实现。

分布式数据库中，分片的数据可以存放在不同节点上，例如，分公司的数据表可作为总公司数据表的一个垂直分片。复制数据时，每个分公司所在的节点是一个主复本节点，同时也作为其它分公司的数据表的从属复本节点。利用 Sybase 复制服务器提供的技术，主复本节点上数据表的修改会被复制到从属复本节点的复制表中。

1.2 ORACLE7复制

ORACLE7的复制结构采用快照技术实现数据复制。快照是主表的完全复制复本或是主表的满足某些选择条件的子集。对远程节点上的数据表可在本地生成一个快照。每次刷新后主表的任何变动都会及时传递到快照中去。

数据的修改也是从原始节点传播到远程节点的复本，但 ORACLE7并不检查数据库日志，它依靠触发器和异步 PL/SQL 远程过程来传播数据的修改。

原始节点上数据的插入、删除和修改都会激发触发器。首先，初始化异步远程调用(RPC)，将过程调用请求加入本地节点上的传播队列，而后在一个独立事务中把过程调用传播到远程节点上。远程节点按被调用过程在本地节点传播队列中的提交顺序执行异步 RPC 事务。

ORACLE7的复制增添了两个新功能：动态所有权和共享所有权。动态所有权在任何时间内只允许数据修改操作在一个节点上执行，共享所有权没有这个限制，多个节点可直接修改所有数据复本。显然，共享所有权很容易引起数据修改冲突。ORACLE7利用前后映象检测冲突，如果一个传播中的前映象记录和远程节点的对应记录不一致，就认为产生了冲突。冲突的解决是通过执行用户自编的子程序使数据库恢复到一致状态。

1.3 Informix 数据复制

Informix 利用数据复制技术提高系统的可用性。存放主拷贝的服务器称为原始站点，动态地接受从原始站点传来的修改的服务器称为第二站点。Informix 用第二站点作为原始站点的紧急备份。在系统正常运行时，原始站点周期性地将逻辑日志传给第二站点，后者按逻辑日志内容修改本站点数据。

如果原始站点送出日志后要等第二站点修改完成的回答才提交事务，这就等同于2PC 协议方式，可保证两站点数据完全一致。若为提高效率，原始站点的事务提交独立地执行，日志传送越频繁则两站点的一致性越好。这时当原始站点发生故障时就可用第二站点自动接替工作，可能有部分事务丢失，但它们的日志都保留在原始站点的日志文件中，待站点修复后再恢复。

2 2PC 协议与数据复制

2PC 协议用于避免在执行提交过程中因节点或通讯故障破坏事务的原子性。2PC 协议分两个阶段：表决阶段和执行阶段。表决阶段，主节点送“准备提交”的信息到所有从节点，从节点收到信息就回答“就绪”(Ready)或“撤销”(Abort)。从节点在回答前把有关信息写入自己的日志中。主节点在发出准备提交信息前也把有关信息写入自己的日志中。执行阶段中，主节点收到的所有从节点的回答是就绪，则向所有从节点发提交命令，否则发撤回命令。从节点收到主节点的最后决定后执行。主节点发最终的提交或撤回命令前将相应记录写入日志，从节点收到最终决定后也将决定写入日志。

2PC 协议已被普遍采用。当系统发生故障时，各节点利用各自相关的日志信息便可执行恢复操作。使用2PC 协议可以保证数据的一致性和事务的原子性。但节点数增加，DBMS 将大量时间用于回答各节点的应答信息，降低了事务的处理效率，延长了事务完成的时间。

数据复制克服了上述困难，但在异步复制中，数据的修改过程与复制过程物理上是分离的两个过程，复制沿不同路径传播就会发生冲突，这样就很难维护数据一致性。复制系统一般采取适当的复制结

构预防冲突的发生，指定一个数据源，数据源所在的节点为主节点，拥有和主节点相同数据的节点为从节点，从节点上的数据是数据源的只读复本。数据的修改只能在主节点上执行，然后传播给从节点，能保持最高程度的数据一致性，而ORACLE7共享所有权就无法达到这一要求，一旦数据库操作失败，已提交的修改就不能撤回。这时，数据库则把解决问题的责任交给了用户，让用户编写子程序解决冲突。Sybase把数据修改置于事务处理中，主节点事务处理结束后执行异步事务传播过程，但主节点事务失败必须由人工干预修改错误。

3 数据一致性

分布式数据库中，数据被存放在多个节点上，维护节点之间的数据一致性成为DBMS的重要责任之一。为了使数据复制满足数据一致性的要求，可以对2PC协议下的可用复本算法（AC算法[4]）加以修改。

多复本控制中，数据重复存放在不同节点上，就每一事务Ti而言，该算法分操作和传播两个阶段。

A. 操作阶段

Ti发wi[x]操作：

1. 给x的主节点a上的xa发wi[xa]操作；

2. a故障，Ti得到超时信号，写操作丢失；

3. a正常，根据xa是否已初始化分情况处理：(1)xa已初始化，先向xa的所有复本发一写标签，然后执行wi[xa]，根据执行结果发出成功或失败回答。若失败，Ti撤回；(2)xa未初始化，当写操作为插入时，a利用此写操作对xa初始化，当写操作为修改时，使xa保持不可用状态，写操作丢失。

Ti发ri[x]操作：(接右上角)

1. 选择就近复本xa。若xa已由事务T0执行过写或复制操作，而不论T0是否成功，xa已初始化；

2. 若节点a正常，则在写或复制xa事务结束后执行ri[xa]操作：(1)a为主节点，则xa为当前最新版本，执行ri[xa]操作；(2)a为从节点，若xa上有写标签，则转向读xa的主节点b，执行ri[xb]。因为x的从节点上的写标签在复制完成后被删去，这样保护Ti读到的x为最新版本。根据执行结果回答成功或失败。若失败则Ti撤回；

3. 若节点a有故障或xa未初始化，事务Ti等不到回答，则Ti发出读x的另一复本操作，若x已无复本可读，Ti撤回。

B. 传播阶段

Ti发Ci操作：

1. 写丢失传播。向主节点再发查询信号，若它已修复，则重新执行Ti。

2. 成功访问传播。向每一成功访问过的节点再发验证信号，若它新失效则Ti撤回，若它正常则Ti提交并发Pi[X]操作，把复制传播给从节点，复制完成后删去该节点上的写标签。若有从节点未收到传播来的复制或复制操作失败，则该复制等待下一次传播到此节点上，节点上的写标签保留。

例 事务T0={w0[x],w0[y],c0}，T1={r1[x],w1[y],c1}，T2={r2[y],w2[x],c2}，各事务的经历图如：

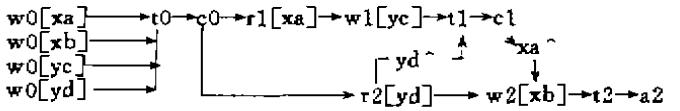
T0=w0[x]→c0

w0[y]↗

T1=r1[x]→w1[y]→c1

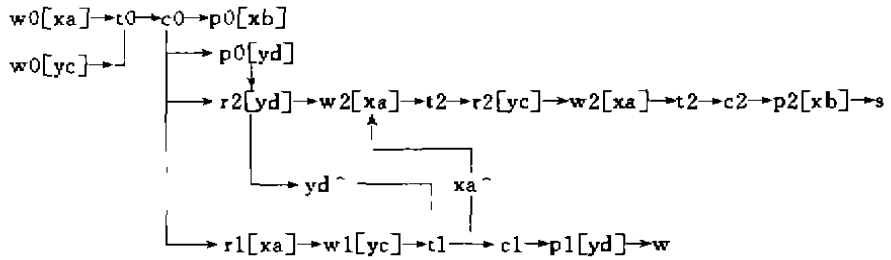
T2=r2[y]→w2[x]→c2

假设yd^→t1→xa^→t2。2PC协议下，可用复本算法产生如下调度H1，ti为验证开始时间：



H1中，T1发r1[xa]时节点a正常，r1[xa]完成，然后执行w1[y]操作，即向c，d节点发w1[yc]，w1[yd]，此时yd^在t1之前，w1[yd]丢失，w1[yc]正常完成，T1进入成功验证阶段。T1向d发出写丢失验证信号，未得到回答，再向a，c发成功验证信号，此时a，c仍正常，验证通过T1提交。T2执行w2[x]时和T1相同：w2[xa]丢失但w2[xb]顺利执行。进入验证阶段后，T2向a节点发写丢失验证信号，此时a节点已修复，因新修复节点未修改，写丢失验证未通过T2撤回。

相同条件下，利用数据复制算法产生调度H2，ti为传播阶段开始时间：



H2中，T0执行 w0[x]，w0[y]，分别发一写标签给 b，d 节点，顺利执行并提交，复制操作结束后删去 xb，yd 上的写标签。T2写 xa 时，因 a 发生故障写操作丢失，在进入写丢失传播阶段再向 a 发出查询，a 已修复重新执行 T2。此时 T2读 yd 的操作因 d 故障而失败，T2发出读 x 的另一复本操作 r2[yc]，yc 是主节点，读操作完成发 w2[xa]信号，同时发写标签给 b 节点，T2进入成功访问传播阶段，向 a，c 节点发成功验证信号，a，c 正常，最终 T2顺利提交，xb 的复制操作完成，b 节点上的写标签被删。执行 T1时，r1[xa]和 w1[yc]顺利完成，进入成功访问传播阶段，因 d 故障，yd 上复制操作失败，但 T1顺利提交，p1[yd]等待下一次传播，d 节点上的写标签保留。

下面试证明该算法的正确性，首先该算法有如下7条性质，其中 ti′ 为写丢失传播开始时间，ti 为成功访问传播开始时间，符号 ^ 和～分别表示数据的失效和创建。

1. 对每一 Ti，oi＜ti′＜ti＜ci＜pi，oi 代表 Ti 的写操作或读操作，pi 为复制操作。

2. SG(H)无环。

3. 若 Ti→Tj，在 SG(H)中，有 ti＜tj′。

4. xa～＜ri[xa]。

5. xa～＝ti 或 xa～＜wi[xa]。

6. 若 H 中有 ri[xa]或 wi[xa]，则 ti＜xa^；若 H 中有 pi[xb]，则 xb～＜pi[xb]。

7. Ti 读 x 的一个非 xa 复本(a 为数据 x 的主节点，b 为从节点)，则 xa^＜ti 或对任何 wj[xa]，ti′＜wj[xa]

命题 H 是算法产生的经历，Ti 读 x 的一个复本但不是 xa(a 是数据 x 的主节点)，则 xa^＜ti 或 ti＜xa～

证明：设 Ti 读 xb，由性质7，有 xa^＜ti 或 ti′＜w0[xa]，其中 T0是初始化 xa 的事务。前者与命题结论一致，所以只需证后者成立。

先设 w0[xa]在 H 中，可证 ri[xb]＜w0[xa]，否则有性质3，t0＜ti′，可推出 t0＜ti′＜w0[xa]＜t0，矛盾。从 ri[xb]＜w0[xa]可推出 ti＜t0＝xa～。

再设 T0未写 xa，与 T0是初始化 xa 的事务前提矛盾，命题得证。 □

比较 H1和 H2，不难看出，2PC 协议为维护数据的一致，只好撤回一些事务，这样便降低了事务处理速度，使事务提交成功率下降。数据复制只要节点正常，事务一般都能成功提交，但复制操作沿着不同路径传播时，很容易引起冲突，导致数据库操作失败。

参 考 文 献

[1] Doug Stacey, Replication: DB2, Oracle, or Sybase? Database Programming & Design, Vol. 7, No. 12, 1994

[2] Fred Schuff, Replication: Answering the Call, Database Programming & Design, Vol. 7, No. 4, 1994

[3] I. W. Draffan 等, Distributed Database, Cambridge University Press, 1980

[4] P. A. Bernstein 等, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley Publishing Company, 1987

[5] 王珊、孟小峰、朱鸿雁，复制器技术的发展状况和展望，计算机世界专题综述，1994. 9. 7

[6] 施伯乐、丁宝康、楼荣生，数据库系统导论，高等教育出版社