第21卷 第5期 计 算 机 工 程 1995年9月
Vol.21 № 5 Computer Engineering Sept. 1995

·软件技术·

开放式犯规提交协议

董小洁 夏宽理
(复旦大学计算机系 上海 200433)

摘要 推出了可以忍受任意数目的可信结点的失职和不可信结点的失职或犯规的一族2PC协议，它们保证参与事务的(至少是)可信结点一致地终止事务，因此称为开放式犯规提交协议。

关键词 失职 犯规 开放式系统 协调者 参加者

Open Commit Protocols Against Commission Failures

【Abstract】 It this paper, a family of 2pc protocols are proposed, which can tolerate any number of trusted nodes' ommission failures and nontrusted nodes'commission and ominission failures. They can ensure that (at least) the trusted nodes eventually terminate the transaction in a consistent manner.

【Key words】 ommission failure / commission failure / open systems / coordinator / participant

为了保证分布式事务的原子性，出现了2PC协议。协议是用来协调事务的终止，使参与事务的结点要么全部提交，要么全部异常中止。协议的复杂性往往依赖于结点的失败模型，导致结点不执行正确算法所定义的行为称为结点的失职，而导致结点执行算法定义以外的动作称为犯规。如结点崩溃为失职，而结点发送错误消息为犯规。

大多数提交协议基于两点假设：(1)结点只有失职；(2)结点最终从失败中恢复。满足上述假设的称为稳健结点。在2PC协议中，只有所有参加者均为稳健的结点才能保证事务一致地终止。

然而，上述假设在开放式分布系统(ODS)中是不现实的，ODS的开放性在于任何可按协议通信的结点均可连接到ODS上，结点的可靠性有完全不同的特点，据此，可把结点分为可信(trusted)的和不可信(nontrusted)的：

在可信结点上只有失职。可信结点在失败时暂停，不会偏离算法，它可以从失败中恢复。

在不可信结点上，可能有失职和犯规，还可能有永久性失败。

本文中讨论的协议是针对ODS，称为开放的2PC协议。它假设只有不可信结点才可能是不稳健结点，还对网络作了容易实现的假设。开放的2PC协议容许任意数目的可信结点的失职和不可信结点的失职或犯规，以保证下列要求得到满足：

(R_1)若事务的参加者均为稳健的，则它们最终会终止事务，否则，(至少)每个可信的参加者最终将终止事务。

(R_2)若事务的参加者均为稳健的，则它们一致地终止事务。否则，(至少)所有可信的参加者一致地终止事务。

R_1和R_2是为了"保护"可信结点，R_1防止可信结点因不可信结点的永久失败造成阻塞，R_2防止不可信结点在事务终止时破坏可信结点的数据一致性。*

1 模型

分布式系统由计算结点和通信网组成，系统中结点合作处理事务，事务分组为事务序列(T-

* 董小洁 女，复旦大学计算机系软件专业，在读硕士研究生

收稿日期：1994-08-12，修回日期：1994-10-24

序列)，是顺序执行的事务集，每一序列逻辑上与进程树(process tree)相联系，后者模型化 T-序列执行。进程树的边代表进程间的客户机／服务器(client／server)关系。进程树的根进程启动 T-序列中的所有事务，称为启动(initiator)进程，在每一事务进程的执行中，每个进程只与它直接的邻居合作，称为上级(superior)，下级(subordinate)。祖先，后辈关系是上级，下级关系的传递闭包。随着时间的变化，进程树可能因为事务生成或摧毁进程而增长或缩减。对于每个事务存在一棵执行树(execution tree)，它由进程树中结点和边的子集构成。例如，图 1 中的事务 T_2 可能由 P_1，P_2，P_4 和 P_5 进程所组成。

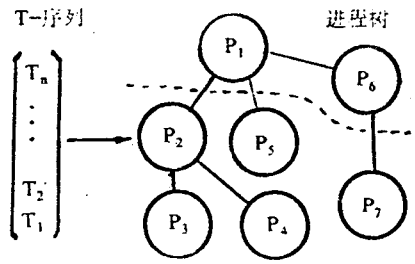


图 1　进程树的例子

在可信结点运行的进程称为可信进程，否则称为不可信进程。在此模型中，进程树中的不可信进程可能有可信的和不可信的下级，而可信进程只能有可信的下级。这种假设减少了终止协议的复杂性，在大多数分布应用中也是现实的。并在假设事务中至少涉及一个不可信进程，即事务树的根总是不可信的。若一个可信进程的上级是不可信进程，则称为入口进程。图 1 中，P_1.P_6 是不可信进程，其它是可信的，因此，P_2，P_5，P_7 上入口进程。

2 开放提交协议

开放提交协议(OC)的基础是推测中止(Presumed Abort)协议，分两个阶段描述:

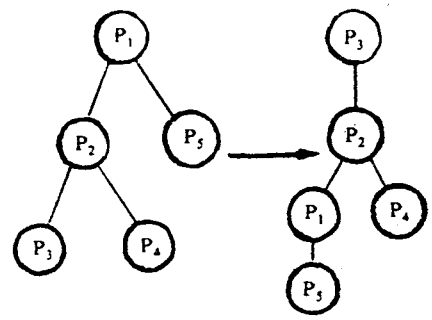
2.1 忍受永久失职

在协议的第一阶段，协调者询问所有参加者是否愿意提交事务。若每个参加者回答 yes，协调者决定提交，否则，异常中止。然后，协调者进入第二阶段，把决定传播给每一个参加者。

愿意提交事务的结点在回答 yes 前，进入预备状态，这时，它不能单方面中止事务而必须等候协调者的决定。这种依赖性产生了 2PC 的阻塞性，即某些失败可能迫使一些结点在失败恢复后才能中止事务。典型的阻塞(blocking)由协调者的失败引起，若它在传播决定前崩溃，每个"预备"结点一直阻塞到它恢复为止。若没有永久的失职，2PC 协议保证阻塞的恢复。但若结点永久失败，则阻塞一直延续，违反了 R_1。

为了避免永久阻塞，可以这样修改协议:(1)协调者功能在可信结点上；(2)恢复时，参加者与协调者直接通信。因为可信的协调者最终必能从失改中恢复，保证了所有参加者收到协调者的决定。在多层次提交协议中，协调功能在启动结点，它是不可信结点，因此，必须提供转移协调者功能到网络中可信结点上的办法。在本论文中对此不作详细讨论。

OC 协议代表了终止树(termination tree)上的操作，终止树是执行树中把协调者功能从启动进程转移到可信进程的重构造。图 2 例示了执行树、终止树的转换。



P_1，P_2：不可信进程，P_3，P_4，P_5可信进程

图 2　执行树和终止树的例子

(1) 正常操作时的 OC

OC 在终止树上的操作，根结点作为协调者。在提交过程中，进程可能处于 3 种稳定的状态:"未知"、"预备"、"在提交"。稳定状态即使在结点失败后仍可以维持。

图 3 表明了 OC 消息流。第一阶段，协调者传送 prepare 消息到下级并等待回答。叶进程收

到 prepare 消息后，决定是否愿意提交事务．若想异常中止事务，发送 no__vote，反之发送 yes__vote．另外，drop__vote 表示只读，说明这个参加者不必参加第二阶段．

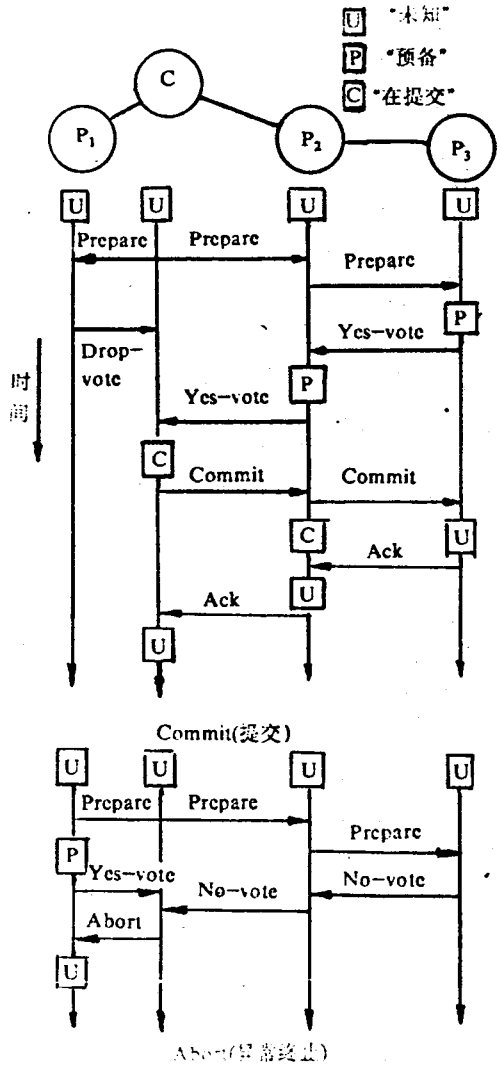


图 3　OC 消息流

当非叶／根进程收到 prepare 消息，传送它的下级并等待选票(vote)，根据它们的选票决定自己的选票．当协调者收到来自所有下级的选票，它决定最后的答案，其决定过程见表 1.

在 OC 第二阶段，协调者向 yes__vote 的下级发送决定．见表 2，进程收到决定时，动作见表 3.

(2) 恢复(recovery)时的 OC

结点失败或发觉上级或下级结点失败时进行恢复．假定每个结定都存在可对其它结点的恢复进程传来的消息进行操作的恢复进程．

作为崩溃恢复的一部分，恢复进程读日志，对处于"在提交"和"预备"状态的事务继续其提交过程，见表 4.

上述算法可能会因永久性失败而导致恢复进程永远"记住"事务．如，一个不可信下级在发送 ACK 前永久失败，则上级结点的恢复进程不断发送 commit 给它．这种情况可能导致大量事务状态信息的积累从而减慢恢复过程．

为避免上述情况，可以给协调者增加恢复计时器(r__timer)来解决．协调者在等待所有 ACK 到达时，若计时器超时，则恢复进程"忘却"(forget)该事务．

同时，为了防止不一致终止，可以在每个协调者上维护一个档案进程(archive process)．这个进程归档所有提交事务的标识符，该事务是在恢复进程收到所有 ACK 前"忘却"的．有了档案进程．恢复进程就可以无二义地回答查询了．至此，讨论了崩溃恢复协议．当进程觉察终止树中的邻近结点的错误时，也要进行恢复，方法与上述类似．

2.2 忍受犯规

上述协议即使在永久失败时仍满足 R_1，R_2，但在有犯规时可能不满足 R_2，如图 4.

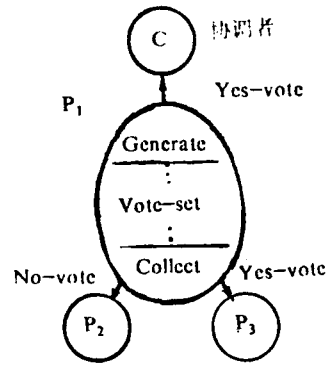


图 4　犯规错

若一个入口参加者的不可信祖先在提交过程中"撒谎"(lie)，则可能破坏 R_2．有两种克服的办法：(1)重构终止树　使得在提交过程中，协调

者与入口参加者直接通信；(2)检测犯规　提交协议增加检测犯规的机制，此机制基于簿记(token)的书架(bookkeeper)概念。

理论上，投票过程可以分为收集(collect)和生成(generate)功能(见图 4)，收集过程把收集到的下级选票传给生成过程，后者决定自己的选票，发送相应消息给上级。两种功能均可能工作失误。

协调者 C 检测犯规的第一步，是在选票消息中，增加选票集。选票集可能出错，必须要求 C 收到的选票集是可靠的，即与 P_2，P_3 发出的是一致的。故引出了簿记概念，来鉴定 P_2，P_3 选票的有效性。

簿记概念用来鉴定可信结点间通过不可信路径交换的消息。一簿记(至少)由簿记类信息(如 yes__vote 或 drop__vote)，事务标识符及簿记生成器标识符组成，具有下列特点：(1)可信进程产生的簿记不能伪造，对它的修改可以被察觉；(2)协调者可以验证可信参加者产生的簿记的可靠性。

簿记的使用保证了 C 收到的选票集是可靠的，但仍可能不完全。如 P_1 可能扣留 P_2 的 no__vote，而传送 P_3 的 yes__vote，也导致 C 的错误决定，为了检查选票集的完整性，引入书架概念。

每一 T-序列与一个书架对应，书架位于可信结点，并保存此序列的入口参加者的资料(book)，书架协议保证要么书架知道参与当前事务的所有入口进程，要么对此序列一无所知。对书架实现在此不作详细讨论。

OC 有 3 种不同的开放协议：直接(Direct)--DOC、半直接(Semi__Direct)-SDOC 和间接(Indirect)--IDC。在 DOC 中，协调者在整个提交过程中与入口参加者直接通信；SDOC 中，第一阶段协调者／入口进程的通信是间接的，第二阶段是直接的；IOC 中均为间接的。3 种协议功能相同，只是处理犯规有所不同。使用哪种较合适主要取决于应用的通信环境。IOC 对面向连接的环境最合适，而在无连接环境中，DOC 或 SDOC 较好。

表 1　第一阶段后一部分的动作

接收的消息	接收者：动作
无 no__vote 且至少有一个 yes__vote	非叶／根进程：向日志中写入预备记录，进入"预备"状态，向上级返回 yes__vote 协调者：决定为提交，向日志中写入提交记录，进入"在提交"状态
皆为 drop__vote	非叶／根进程：向上级发送 drop__vote，并立即"忘却"该事务； 协调者：整个事务是只读的，立即"忘却"该事务
存在 no__vote	非叶／根进程：向上级返回 no__vote，向所有返回 yes__vote 的下级发送 abort，并清除事务的局部影响 协调者：决定为异常终止

表 2　第二阶段协调者动作

发送的决定	动　作
commit	向 yes__vote 的下级发送 commit 并等待确认消息。收到所有下级的 ACK 后，向日志中写入"结束"记录，进入"未知"状态
abort	向下级发送 abort，并异常终止该事务

表 3　第二阶段非协调者动作

收到的决定	动　作
commit	叶进程：提交事务，进入"未知"状态，向上级返回 ACK 非叶／根进程：进入"在提交"状态，向日志中写入提交记录，向"预备"的下级发送 commit，并等待 ACK，收到所有 ACK 以后，向日志中写入结束记录，进入"未知"状态，并向上级返回 ACK
abort	把 abort 传播给下级(若有)，删除事务，向日志中写入异常终止记录，进入"未知"状态

表 4　恢复过程动作

对"预备"状态的事务	对"在提交"的事务
入口结点的恢复进程 → 周期的协调者查询	
不可信结点的恢复进程 → 周期的协调者查询	结点的恢复进程 → 周期的所有下级查询
可信非入口结点的恢复进程 → 查询上级 并根据协调者或上级返回的 commit 或 abort 消息终止事务	收到所有的 ACK 后，向日志中写入结束记录并"忘却"该事务

3 结束语

传统的 2PC 协议假定事务中的所有结点是稳健的，而 ODS 中，这种假设是不现实的。因为这里有可信结点和不可信结点。为此，本文提出了新的 2PC 协议，它们能忍受永久的失职和犯规。在无犯规时，保证所有稳健参加者最终一致地终止事务；在发生失职和犯规时，保证所有可信结点最终一致终止。这些协议关于网络的假设是可以在 ODS 中得到满足的。

参考文献

1 Rothermel K., Pappes.Open Commit Protocols Tolerating Commission Faiures.ACM Transactions on Database Systems, 1993-6, 18(2):289-297

2 Garcia Molina H, Abbott. R K. Reliable distributed database management. Proce edings of the IEEE 75,5.1987:601-602

3 Lampson B.Atomic transactions.Lecture Notes in Computer Science, 1981: 371-376

4 翟兆荣.分布式数据库.上海：华东计算技术研究(内部)

5 陈建荣，严隽永.分布式数据库设计导论.北京：清华大学出版社

6 关荣春等译.分布式数据库原理和系统.北京：水利电力出版社