联动北方
LANDING

企业级 IT 运维宝典系列

联动北方 出品

目　录

第 9 章 如何用好全局事务

全局事务是由资源管理器管理和协调的事务，可以跨越多个数据库和进程。事务管理器一般使用 XA 二阶段提交协议与“企业信息系统”(EIS) 或数据库进行交互。

9.1 什么情况下使用全局事务

一般当一个事务需要跨越多个数据库的时候，需要使用全局事务。例如，一个事务中可能更新几个不同的数据库。对数据库的操作发生在系统的各处，但必须全部被提交或回滚。此时一个数据库对自己内部所做操作的提交不仅依赖本身操作是否成功，还要依赖与全局事务相关的其它数据库的操作是否成功，如果任一数据库的任一操作失败，则参与此事务的所有数据库所做的所有操作都必须回滚。

在一个涉及多个数据库的全局事务中，为保证全局事务的完整性，由交易中间件控制数据库做两阶段提交是必要的。但典型的两阶段提交，对数据库来说事务从开始到结束（提交或回滚）时间相对较长，在事务处理期间数据库使用的资源（如逻辑日志、各种锁），直到事务结束时才会释放。因此，使用典型的两阶段提交相对来说会占用更多的资源，如果网络条件不是很好，如低速网、网络颠簸频繁，情况会更为严重。

9.2 本地事务的优缺点

本地事务容易使用，但也有明显的缺点：它们不能用于多个事务性资源。例如，使用 JDBC 连接事务管理的代码不能用于全局的 JTA 事务中。另一个缺点是局部事务趋向于侵入式的编程模型。

9.3 Tuxedo 对事务的控制与管理

当客户端连接到 Tuxedo 并创建一个全局事务时，TM（Transaction Manager 事务管理器）就会在公告板（BB）里面创建一个事务，由 TMS 向 GTT（Global Transaction Table, 全区事务表，里面包含当前事务的状态信息）中插入一个条目，然后分配一个 GTRID（Globle Transaction Identifier 全局事务标识符）来对该事务进行跟踪。

Tuxedo 的事务管理由 TMS 完成，TMS 把各种 RM 接入到 Tuxedo 中的分布式计算中来，并对 RM 中执行的事务进行跟踪和两阶段提交。

Tuxedo 对事务的管理工作主要包括：创建 TMS、创建 TLOG、运行时事务的监控和迁移。每一个在 Tuxedo 中用到的 RM，都需要创建一个专用的 TMS，否则无法在 UBBCONFIG 文件中调用。

创建 tms 的命令为：buildtms.这个命令需要从 RM 文件中读取信息，包括 RM 名、XA Switch 名，以及 XA 支持库。

为了恢复全局事务，TMS 使用 TLOG 来记录事务日志。在每台 Tuxedo 主机上，只需创建一个 TLOG 文件，它就会被这台主机上所有的 TMS 实例共享使用。如果一个全局事务还没有完成，就会在 TLOG 文件中占用一个分页的空间（512KB），事务完成之后，它在 TLOG 中的记录被自动删除。

全局事务中，如果一个事务在提交前失败，在事务超时以后，TMS 会把它的状态从 TMGACTIVE 改变为 TMGABORTONLY，在 Tuxedo 下一次进行健康检查的时候，会把它从 GTT 中清除。

另外当 Tuxedo 检测到只有一个 RM 参与到分布式事务中时，TMS 则会略去第一阶段是的事务征集过程，直接进行事务的提交或者回滚。

9.4 常用事务相关的函数

为了界定全局事务，Tuxedo 除了支持标准的 TX 接口外，还提供了一套自己的事务接口，其基于 TX 接口的包装：

1. tpopen()

这个函数被服务进程和 TMS（事务管理器）调用，用于建立和 RM（资源管理器，一般为数据库）的连接。连接信息由服务进程组的 OPENINFO 参数提供。

服务进程和 TMS 在启动的时候，通常会自动回调 tpsvrinit()函数，tpopen()通常在 tpsvrinit()函数中被调用。连接失败的时候返回值为-1，并把错误号保存在全局变量 tperrno 中。

2. tpclose()

这个函数在服务进程和 TMS 的析构函数 tpsvrdone(3c)中被隐含调用，用于关闭一个 RM 的连接，关闭信息由进程组的 CLOSEINFO 参数提供。

3. tpbegin()

该函数的功能是开始一个全局事务，并分配一个 GTRID(全局事务标识符)来对它进行跟踪。

4. tpcommit()

该函数的功能是提交一个全局事务，提交成功时返回 0，失败时返回-1。

提交失败时，可能把 tperrno 设置为 TPETIME、TPEABORT、TPEPROTO、TPEHAZARD、TPEHEURISTIC 或 TPEINVAL。其中：

- TPETIME 表示事务已经超时，状态未知，可能是已经提交，也可能是已经回滚了；
- TPEABORT 表示某个 RM 不能提交它的局部事务；
- TPEPROTO 表示协议错误，即调用点不在一个有效的事物上下文中，比如事务的提交者不是事务的初始者或者提交的事务根本就不存在；
- TPEHAZARD 表示由于某些失败的因素，全局事务已经启发式完成；
- TPEHEURISTIC 表示由于启发式的决策，部分 RM 提交了事务，部分 RM 回滚了事务；
- TPEINVAL 表示函数调用的参数设置不对。

5. tpabort()

回滚一个全局事务

6. tpsuspend()

该函数功能为挂起一个全局事务。当某些对 RM 的操作不想纳入当前的事务上下文中的时候，可以在调用点之前先挂起事务，当前事务完成后，再恢复事务。

7. tpresume()

恢复一个被挂起的全局事务。

8. tpscmt()

该函数的功能是设置提交控制参数 TP_COMMIT_CONTROL 的值。

9. tpgetlev()

通过该函数的返回值来判断当前的调用点是否处在全局事务中。如果返回值是 1 表示当前调用点正处在全局事务中，0 表示不在全局事务中。

9.5 数据库连接

9.5.1 TMS 介绍

Tuxedo 事务管理器（TMS）必须跟踪分布式事务处理的整个流程，记录足够的信息以便在任何时候进行提交或回滚，因此 TMS 使用事务日志文件（TLOG）来记录跟踪信息，同时为了区别系统中同时进行的不同事务处理流程，TMS 又为不同的事务处理分配了一个全局事务编号（GTRIDs）。

在事务处理的不同阶段，TMS 将执行不同的动作如下表：

阶段	TMS 动作
应用程序启动一项事务处理	为事务处理分配一个全局事务编号（GTRIDs）
启动事务处理的进程与其它进程通信	跟踪这些参与事务处理的进程
事务处理访问 RM	将相应的 GTRIDs 传递给 RM，这样 RM 就可以监控哪些数据库记录被该事务处理存取
应用程序标记一项事务处理将被提交	按两步提交协议执行事务
应用程序取消事务处理	执行回滚操作
有错误发生	执行回滚操作

表 9-1

9.5.2 XA 模式与 NO-XA 模式

XA 就是 X/Open DTP 定义的交易中间件与数据库之间的接口规范（即接口函数），交易中间件用它来通知数据库事务的开始、结束以及提交、回滚等。XA 接口函数由数据库厂商提供。

NO-XA 应用服务器不需要参与事务管理。只针对单一事务资源。不能跨越多个事务资源。

9.5.3 与各种数据库的连接

Tuxedo 可以和所有的有标准 XA 接口的 RM 连接,目前几乎所有的关系型数据库和消息队列产品，都支持标准的 XA 接口。Tuxedo 和各种数据库相连,都需要配置一个重要的文件 RM。

RM 文件包含所有的资源管理器的入口，它们被 Tuxedo 应用访问，RM 文件在 $TUXDIR/udataobj 目录下。

以 Oracle 数据库为例：

1. 操作系统准备工作

如果 Tuxedo 连接的数据库不在本地的话,需要安装 oracle 客户端。

2. Oracle 数据库中的准备工作

Sysadmin 登录数据库

执行脚本

```
SQL>@$ORACLE_HOME\rdbms\admin\xaview.sql
```

示例 9-1

赋权限给 public 用户

```
SQL>grant select on v$xatrans$ to public with grant option;
SQL>grant select on v$pending_xatrans$ to public with grant option;
SQL>GRANT SELECT ON DBA_PENDING_TRANSACTIONS TO public;
```

示例 9-2

3. .profile 文件的设置,需要设置 ORACLE_HOME 并修改 PATH

```
ORACLE_HOME=/u01/app/oracle/product/10.2.1/client
export ORACLE_HOME
PATH=$PATH:$ORACLE_HOME/bin
export PATH
```

示例 9-3

4. 修改 RM 文件

如果使用的不是 COBOL(Common business Oriented Language)开发的程序, Oracle_XA 的值不需要改变,否则需要修改为:

```
Oracle_XA:xaosw:-L${ORACLE_HOME}/lib
${ORACLE_HOME}/precomp/lib/cobsqlintf.o  -lclntsh
```

示例 9-4

5. 创建 tms 文件

在 TUXAPP 目录下创建文件 TMS_ORA10G, Tuxedo 通过 TMS_ORA10g 与 ORACLE 数据库采用 XA 协议进行通讯

```
buildtms -o $TUXAPP/TMS_ORA10g -r Oracle_XA
```

示例 9-5

6. 修改 UBBCONFIG 文件

在*GROUPS 中添加:

```
OPENINFO="ORACLE_XA:Oracle_XA+Acc=P/scott/scott+sqlNet=ORCL+SesTm=100+LogDir=.+MaxCur=5" TMSNAME="TMS_ORA10g" TMSCOUNT=2
```

示例 9-6

9.6 全局事务的使用规则

全局事务的使用遵守两阶段提交协议，另外在事务控制问题上还有以下几个地方需要注意：

9.6.1 谁发起谁结束

全局事务的发起和结束可以是中间件应用的前台，也可以是后台。在事务的控制上应遵循谁发起发起事务，谁就结束的原则。

我们知道在 Tuxedo 中事务既可以在前台程序中发起，也可以在后台程序中发起。无论放在前台还是放在后台都有其优缺点。事务放在前台增加了网络传输的流量，但是可以保证异常情况下前后台操作的一致性，事务放在后台可以减少网络流量，但是对于异常情况下前后台操作的一致性很难保证。

通常采用的是事务放在前台程序中。但是无论放在前台还是后台，都要遵循谁发起，谁结束的原则。

9.6.2 不允许嵌套

Tuxedo 不支持嵌套事务处理，即发起者在调用 tpbegin()和 tpabort()或 tpcommit()之间不能再调用 tpbegin()开始一个新的事务处理，也不能再开始一个本地事务。

9.6.3 处理好超时

Tuxedo 应用系统的事务超时控制很重要，不设置超时时间对系统来说可能会引发灾难。影响 Tuxedo 全局事务的超时主要有三种：

一个是在代码中的 tpbegin()超时（值为其参数）T1，它控制整个事务的完成时间；

第二个为数据库 XA 连接的超时（配置文件中的 open_info 中的 SesTm ）T2，它控制同一连接中对于分布式事务锁的等待超时时间；

还有一个为数据库中等待数据库对象分布式事务锁释放的超时时间（_dirstributed_lock_timeout）T3；

这三个超时共同作用并且有如下的关系：

```
T1<T2<T3
```

示例 9-6

9.7 事务挂起的问题

在使用全局事务的情况下，可能会遇到事务挂起的情况，即：SERVER 进程还在，但是无法响应请求。在服务对应的日志文件中总是报“当前的进程已经在一个本地事务中了”的错误，这时只有重启该 SERVER，才能排除故障。其实这是一个事务控制的问题，它产生的根本原因是在开启全局事务前，该 SERVER 执行了一个本地的事务并且没有提交或回滚。在程序代码上表现为如下三种原因：

（1）在调用该 SERVER 的某个带 DML 语句的 SERVICE 前没有加 tpbegin。

（2）在调用 tpbegin 后没有判断返回值。

（3）tpbegin 后的 tpcall 服务调用没有判断返回值，在全局事务超时后没有能够及时的退出后续的调用，导致下一个 tpcall 产生了一个本地事务。

此外，还会有一类比较特殊的问题，那就是 TMS 服务挂起的问题，利用 tmadmin 中的 pq 命令发现 TMS 的队列中有很多请求存在，在这种情况下，一般只能等待其执行完成，情况严重时，则需要调整相应的数据库参数，在 ORACLE 中该参数应该设为 max_commit_propagation_delay>=90000（max_commit_propagation_delay 表示 scn 在 sga 里面刷新的最大时间，单位为 0.01 秒）。

在实际的运行维护中，我们发现即使把数据库参数调整了，有时还会有 TMS 挂起的故障发生。针对这件事情我们专门作了研究，TMS 之所以挂起是因为 TMS 在等待数据库中 DX（distributed execution lock）独占锁的释放，这种独占锁加锁的对象一般都是 ORACLE 的系统对象，而这种独占锁的产生一般是在全局事务中执行着 DML 语句，当这种 DML 语句执行比较慢时，就会引起 TMS 的锁等待现象。此外，我们还发现，一般的查询语句是不会产生锁的（OPS 的 lm_lock 锁除外），但是如果将查询放入一个全局事务中时，它就会产生一个共享的 DX 锁，如果这个在全局事务中的查询的调用是通过 ORACLE 的工具包 DBMS_SQL 来进行的话，那就会产生一个 DX 的排他锁。

基于这些结果，我们建议在程序的开发过程中要遵循能不用 XA 全局事务的情况下就不用，能将查询放到事务之外的就不要放到事务之内，能不用 ORACLE 工具包进行开发的就不要用。