分布式事务处理的若干关键技术*)

The Key Technologies in Distributed Transaction Processing

沈镇林　姚国祥

(暨南大学网络中心　广州510632)

Abstract　The key technologies in distributed transaction processing are introduced and their implementation, their research situation and some related problems are discussed.

Keywords　Distributed transaction processing, Database, Two phase commit, Replication server, Common object request broker architecture

保证本地/异地访问控制和同构/异构数据库管理的数据一致性和事务完整性是分布式联机事务处理的关键因素之一。

在数据库系统中,事务是维护数据一致性的单位,它使数据库从一种旧的一致状态转变为一种新的一致状态。在每个事务结束时,数据库系统总是保持一致性。事务分为单节点数据库事务、多节点同构数据库事务和多节点异构数据库事务。

1. 单节点数据库事务

两层半体系结构是两层体系结构逻辑上的扩充。由于数据库系统变得更加先进,因此将过程扩充部分引入SQL标准,允许程序设计者编写过程逻辑,并在服务器上执行它。如Sybase中的这些过程语言扩充用Transact-SQL语言表示,并完全允许在服务器上开发健壮的应用程序。

通过事务控制语句、封锁机制和事务日志来实现对事务管理以及数据一致性约束。

2. 两级事务模型

多数据库事务处理是多数据库系统的核心问题,多数据库系统具有自治性、异构性和分布性的特征,使得多数据库事务处理与传统事务处理有很大的不同。多数据库中存在两种事务,全局事务和局部事务。如果将集中数据库中多级事务模型简化为两级事务模型,那么就可将其运用于多数据库中。下面就多节点同构数据库事务和多节点异构数据库事务处理、现有的相关技术进行分析。

2.1　远程过程(RPCs)

远程过程是从一个数据库的存储过程中调用另一个同构数据库上的存储过程。例如,数据库服务器(margin_1)上的存储过程sp203调用数据库服务器(margin)上的存储过程rpc203:

```
begin tran pre_p_d
...
 execute @ retstat=margin.mgdb.dbo.rpc203 @ contract
...
 If @ retstat<>0
  begin
    rollback tran
    return-999
    end
 else commit tran
```

当故障发生时,在一个事务中被调用的远程过程是不能够恢复(rolled back)的。因此,这种远程过程在实际分布式事务处理系统中是无法确保数据一致性和事务完整性的。

2.2　两阶段提交(2PC)

两阶段提交在出现系统故障或网络故障时能提供一种故障恢复机制。在基本的两阶段提交协议中,包括准备、提交两个阶段和故障处理。

(1)准备阶段　全局协调节点向参与节点发出准备请求,参与节点通过发出"准备好"、"只读"、"失败"响应请求,并且每个参与节点完成如下过程:请求后续节点作准备、分配事务提交所需的资源、记录事务引起的数据库的各种改变等。准备好后,参与节点等待"提交"或"回退"信号而不会单方面作出是否提交或回退的决定。如参与节点未能成功地完成"准备"请求,此节点释放当前为事务准备的资源并且回退事务的本地部分再向请求节点发"失败"信号。同时参与此分布事务的其它节点也回退事务,这就保证了全局数据库的数据完整性。

(2)提交阶段　提交某分布式事务的第二阶段即为提交阶段。此分布式事务引用的所有节点必须已经

*)本文研究得到国家计委高新技术重点项目资助。沈镇林　副教授,从事计算机网络、数据库技术、信息系统方面的研究。

保证有必要的资源来提交事务后才能进入提交阶段。在这个阶段，所有的节点参与事务的提交即提交此事务的本地部分。提交完成后分布式数据库的所有有关节点上的数据相互间是一致的。

(3)故障处理　在两阶段提交期间总有可能出现故障，事务两阶段提交机制可以从故障中恢复到提交前的全局数据库的状态。

(4)两阶段提交在外汇交易系统中的应用　以笔者参与开发的外汇交易系统为例，外汇交易系统由一个中心数据库服务器和若干个会员数据库服务器组成。交易信息包括：买卖外汇申请、报价信息、成交信息、市场行情等。我们以其中的"更新报价信息"模块为例，说明如何更新中心数据库服务器和各会员数据库服务器报价表(curr_rate)上的报价数据(币别 currency、买入价 buy_rate、卖出价 sel_rate)，下面程序利用微软的 MSDTC 分布式事务处理技术和 C 语言实现同时更改分布在两个数据库服务器上相关表 curr_rate 的 currency、buy_rate 和 esel_rate：

```
login=dblogin();//初始化登录结构
hr=Dtc Transactionmanager(
//获得 ItransactionDispenser 接口指针
NULL, //pszhost,
NULL, //pszTmName,
IID_ItransactionDispenser, //REFIID rid,
0, //DWORD dwReserved1,
0, //WORD wcbReserved2,
0, //void FAR * pvReserved2,
(void**)&pTransactionDspenser //void**ppvObject);
If(FAILED(hr))
{printf("DtcGetTransactionManager failed:%x\n",hr);
   exit(1);
}
//建立到在 mgserver0上的 mgdb0中心数据库的连接
LogonToDB1(&dbproc_mgserver0,&gSrv0);
//建立到在 mgserver1上的 mgdb1会员数据库的连接
LogonToDB1(&dbproc_mgserver1,&gSrv1);
//初始化分布式事务
hr=pTransactionDspenser->BeginTransaction(
NULL,
ISOLATIONLEVEL_ISOLATED,
ISOFLAG_RETAIN_DONTCARE,
NULL,
&pTransaction)
If(FAILED(hr))
{printf("Begin Transaction failed:%x\n",hr);
   exit(1);
}
//列举数据源
Enlist(&gSrv0,dbproc_mgserver0,pTransaction);
Enlist(&gSrv1,dbproc_mgserver1,pTransaction);
//生成 SQL 语句
sprintf (Sqlstatement," update curr_rate set buy_rate =:
        buy_rate, sel_rate =:sel_rate where currency =:
        currency);
//在两个数据库服务器上执行数据 buy_rate 和 sel_rate 的
  更改
ExecuteStatement (&gSrv0, dbproc_mgserver0, Sqlstat-
  ement);
ExecuteStatement (&gSrv1, dbproc_mgserver1, Sqlstat-
  ement);
//提交事务
hr=pTransaction->Commut(0,0,0);
If(FAILED(hr))
{printf("pTransaction->commit()failed:%x\n",hr);
   exit(1);
}
```

(5)存在问题　两阶段提交(2PC)服务能保证在保持场地自动的同时实现分布式数据更新的一致性。但是，由于两阶段提交(2PC)在系统效率和可用性两方面都存在缺陷，一方面与单节点事务相比，2PC 的分布事务太慢；另一方面一旦参与分布事务的某一个结点下适时地失败，会使其它结点无法继续处理数据，从而降低了系统的可靠性。复制服务器有效地解决了 2PC 存在的问题。

2.3　复制服务器(Replication Server)

首先，从效率上看，通过数据在多个节点的副本，使某节点对该数据的访问无需通过网络远程去获取，而且复制服务器的异步 RPC 技术使原来依赖于2PC 技术的分布式事务能快速有效地进行；从可用性看，复制服务器通过 Store-Forward 技术解决了原有系统对网络及远程节点可用性的依赖，在网络或远程节点出现故障时本地正常提交，复制服务器监听各种状态，一旦网络重新连通或远地节点恢复运行，则能将事务传递到远程节点进行同步。Replication Server 在整个 Client/Server 网络的异构平台上同步复制数据，它赋予了高度的本地自治能力和灵活性。

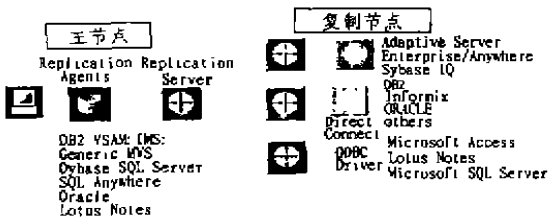

图1　复制服务器

(1)复制原理：1)客户应用修改主节点数据库中的数据项；2)Replication Agent 读取主节点数据库中变化信息；3)Replication Agent 将变化信息传到 Replication Server；4)Replication Server 将 变化信息传播到复制节点中的数据库。

(2)存在问题　异构数据库的透明访问、容错能力和负载平衡能力等问题日益受到人们的关注，而服务器复制技术在这一方面上有不足之处，分布式对象标准的出现，提供了解决问题的方法及实现技术。

3. 分布式对象标准

在计算模式向 Internet 计算迁移的过程中，首先必须解决以下问题：

·资源定位：应用系统可以方便地知道和使用网络系统中可用资源，却不必知道资源的分布位置和多种

多样不同的使用方法。

·代码重用和移值：同一功能代码可在不同硬件和操作系统平台运行和用于构架软件。

·软件互操作：使用不同语言开发的程序可互相调用。

·遗留(Legacy)系统的利用：已经建立并发挥作用的系统能够迁移到网络计算环境并与新开发的系统集成。

·屏蔽异构网络环境下编程的复杂性：使开发人员能专注于应用领域的问题和编程模型而不必关注网络细节。

分布对象计算模式下，构成应用系统的对象分布于网络中，构成整体的应用系统。对象是构成分布对象计算系统的基本单元，封装了网络计算资源。网络中存在大量的对象，对象之间的通信以及对象的管理成为分布对象计算的主要问题。必须有一个支持对象间通信的基础结构和管理对象的机制，对象管理机制使网络中的对象以简单的方式被应用开发人员使用，对象间通信的基础结构使对象间以简单的方式互相操作。

已经有成熟的分布对象标准，用来实现分布式的计算。例如：Microsoft 提出的 COM/DCOM，OMG 定义的 CORBA 以及 Sun 的 JavaBean。这些标准都可以很容易地将商业逻辑封装起来，分布在网络上进行访问。例如：可以通过 DCOM 访问分布在网上的 ActiveX 应用服务器，利用 IIOP 访问 CORBA 对象，利用 RMI 访问 JavaBean。而且许多厂商提出的技术可以让这些对象互相访问，不仅仅局限于一种对象。CORBA 完全可以将各种对象集成。

SYBASE 的 Jaguar CTS 和 Microsoft 的 MTS 就可以让各种对象混合使用，它们组合了 TP Monitor 和 ORB 的功能，并且支持广泛的分布式对象模型标准，使 Web 应用开发方法与传统技术有了本质区别，可以实现实时的网上交易，而不仅限于传统的客户/服务器环境，可以用 Java 访问放在 Jaguar 和 MTS 中的各种对象。正因为这些分布式对象和通讯协议的成熟，使得浏览器到应用服务器之间可以使用新的对象间的通讯协议，而摆脱 HTTP，CGI 的束缚。

4. 下一代的 Web—Object Web

CORBA 和 IIOP 将成为 Object Web 的主宰。浏览器和 Web 服务器中有了 CORBA 的支持，这样它们之间就可以使用 IIOP(CORBA 的 ORB 之间的通讯协议)进行通讯。因此，浏览器中的 CORBA 对象就可以使用 IIOP 调用应用服务器或 Web 服务器上的 CORBA 对象访问数据库，见图2。

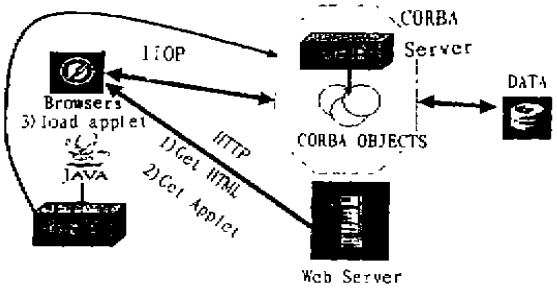

图2 CORBA 和 IIOP

这种新的结构重要的在于可以摆脱 HTTP 这种间断的协议，不再使用 CGI。它可以让浏览器中的 Java Applet 或其它组件通过对象间的访问协议使用位于应用服务器或 Web 服务器上的对象，通过这些对象实现对后台数据库联机地访问，更好地控制每个事务，得到更快的访问速度。除此之外，Java 也扮演了一个重要角色，用 Java 编写 Java 的 ORB 可以动态地从 Web 服务器下载并且运行起来。如同 Java Applet，现在有了 ORBlet，例如 Iona 的 Orbix Web。

参 考 文 献

1 Available at: http://techinfo. sybase. com/cgi/techinfo nsf/docId/ID=47622
2 Available at, http://www. sybase. com/inc/powerline/powerline_97/internet. html
3 Salemi J CLIENT/SERVER COMPUTING WITH SYBASE SQL SERVER. by ziff-Davis Press，1994
4 Sybase 产品指南，Sybase 技术白皮书系列
5 OMG. The Common Object Request Broker Architecture and Specification V. 20，July 1995
6 Otte R，Patrick P，Roy M. Understanding CORBA：the common object request broker architecture. by Prentice Hall PTR，1998