一种基于物理隔离的数据库同步方案

孙业毅,吴国凤
(合肥工业大学 计算机与信息学院,安徽 合肥 230009)

摘 要:电厂的远动数据库有数据量大、更新频率快、准实时的特点。提出了一种在物理隔离环境下内外网远动数据库的数据同步方案。该方案模块划分清晰、通用性好、易于实现,并且不影响内网源数据库和其上的应用系统的正常运行。然后针对SQL Server 2000给出了同步系统实现,经实际运行检验,该方案可行,达到了设计目标。

关键词:镜像;数据库同步;物理隔离
中图分类号:TP393.08 **文献标识码**:A **文章编号**:1673-629X(2007)11-0175-04

A Scheme of Database Synchronization Based on Physical Gap

SUN Ye-yi, WU Guo-feng
(School of Computer & Information, Hefei University of Technology, Hefei 230009, China)

Abstract: Tele-control database of power plant is characteristic of enormous data, fast update frequency and quasi real-time. Proposed a tele-control data synchronization scheme based on GAP device. The scheme's modular division is clearly, easy to be realized, and intranet source database and application system can run normally. Particularly, implemented the synchronization system for SQL Server2000, which is proved to be highly effective, stable and reliable in real missions.

Key words: mirror; database synchronization; physical gap

0 引 言

网络应用越来越广泛,同时面临的安全问题也日益严峻,为保障电厂监控系统与远动系统的网络安全,电厂的计算机监控系统和远动系统与厂级MIS系统连接需要物理隔离[1,2],在电力行业的俗语中这称之为二次系统安全防护。在这里,把电厂计算机监控系统和远动系统所在的局域网称为内网,MIS系统所在的网络称为外网,我们就是要通过物理隔离装置,保证内网的保密性、安全性、完整性和高可用性,阻断任何可能的外网无意或恶意的攻击。在电厂二次系统安区防护中的关键问题就是物理隔离以后,内网的远动数据库中的负荷、电量、电流、电压、频率等数据库中的数据如何像以前一样,继续供MIS系统和远动Web系统访问,将物理隔离所产生的数据交互影响降至最低。为此在MIS系统所在的外网放置一台数据库服务器,作为远动数据库的镜像(以下称之为镜像数据库,内网的远动数据库对应的称之为源数据库),并保持其上的数据在一定的时间内(小于5秒)保持与内网的源数据库中的数据保持一致。MIS系统和外网的远动Web系统转为从镜像数据库获取数据。文中所要解决的问题就是在物理隔离装置将网络隔离的情况下如何将内网的源数据库中的数据及时、可靠地同步到外网的镜像数据库中,并针对SQL Server 2000给出了其详细的实现方案。

收稿日期:2007-01-27
作者简介:孙业毅(1978-),男,安徽寿县人,硕士研究生,主要研究方向为计算机网络;吴国凤,副教授,主要研究方向为数据库、计算机网络。

1 物理隔离装置简介

电力系统专用网络隔离装置是电力二次系统安全防护体系中安全区I、II与安全区III连接的单向专用安全隔离设备,也叫网闸。该装置除采用基本的防火墙、代理服务器等安全防护技术之外,关键采用了"双机非网"的物理隔离技术[3],即装置可以阻断网络直接连接,两个网络不同时连接在设备上;可以阻断网络逻辑连接,即TCP/IP必须被剥离,将原始数据以非网络方式传送;隔离传输机制具有不可编程性;任何数据都是通过两级代理方式完成;具备对数据的审查功能,数据不具有攻击及有害的特性;具有强大的管理与控制功能,并实现了国家电力调度中心的安全防护要求的两种通讯模式:

(1) UDP完全单向通讯方式;

(2) TCP单向数据4字节返回方式。

这里 TCP 单向数据 4 字节返回方式是指正向(由内网到外网)所携带的 TCP 数据包大小不受限制,但反向传输的 TCP 数据包所携带的数据不能超过四个字节,否则被截断。这使得我们可以通过应用程序的编制来完成从内网到外网的数据传输。

2 数据库同步的基本原理

远动数据库的数据量大,一年的数据可达 700M,而通常数据库要保存三年的数据,更新频率快,实时表的刷新在 6~10 秒,所以为减少网络流量,对镜像数据库的同步必须采用增量同步。要实现增量同步,有两个关键问题需要解决。

第一个问题是获得源数据库的变化信息,即源数据库表中的 Insert,Update,Delete 操作信息如何获取,主要有以下几种方法[4]:

(1) 中间件法。在应用程序和数据库之间引入中间件,由中间件完成所有对数据库的修改,在中间件中记录下数据库的变化信息。

(2) 时间戳法。该方法需要为每一个同步表增加一个时间戳字段,以记录每个行的修改时间,同步系统可根据该字段获取变化的行。

(3) 扫描表法。定时对需要同步表扫描,获取两次扫描之间的变化信息,并记录下来。

(4) 触发器法。在源数据库为同步对象创建相应的触发器,当对同步对象进行修改、插入或删除等 DML(Data Manipulation Language)命令时,触发器被唤醒,将变化记录下来。

这四种方法都适用于增量同步的场合,但前二种方法都要求对现有的数据库系统和应用程序做出较大的调整,开发成本高,实施难度大;第三种方法对于记录不是很多的小表是可行的,但对于记录较大的表,则效率低下;触发器法,需要数据库有触发器的支持,对同步表的操作性能会有所影响,除此之外对现有系统无其他影响,因而易于实现。

综上分析,采用触发器法来获取对表的操作信息,首先在源数据库中新建一个跟踪记录表(DBSTRecordsModified),用来存储数据库变化的操作信息,主要有序号、SQL 语句、执行时间三个字段,分别记录操作的顺序号、与操作对应的 SQL 语句和操作的时间,其中序号字段为自增标识列。然后为每张要同步的表创建三个 After 触发器:Insert 触发器、Update 触发器、Delete 触发器,每当对表有更新操作的时候,就会激发对应的触发器,在触发器里生成与该更新操作对应的 SQL 语句,并将其与当前时间一起保存到跟踪记录表中,同步模块在进行同步数据的时候,只要依次取出跟踪记录表中的记录,然后传到镜像数据库上执行就可以完成增量同步。

第二个问题是镜像数据库的初始化,即在增量同步之前如何保证在某一时刻,镜像数据库与源数据库在结构与内容上完全一致。我们假定外网是不安全的网络,那么在数据库遭到破坏的时候,镜像数据库的初始化也是必须的了。解决这个问题的一个难点是不能影响源数据库的正常运行,即不能暂停源数据库的运行,这样会丢失大量的现场数据,针对 SQL Server 数据库系统,可以采用带事务标记的备份与还原技术,在源数据库端将数据库备份成文件,然后再把文件传输到镜像服务器上还原。

3 数据库同步的总体方案设计

应用上述触发器法跟踪记录变化的思想,可以设计和开发出一个独立的、与具体应用联系较少的、通用的数据库同步系统,该系统分为发送端与接收端两个程序,由 VC 6.0 开发。整个系统框架如图 1 所示。

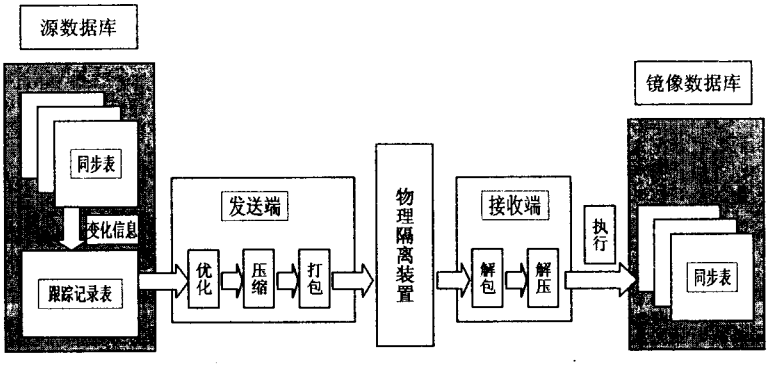

图 1 基于物理隔离的数据库同步系统框图

总体流程为:在发送端,由用户选择和配置需要进行同步的数据库、同步周期,选择需要同步的表,选择需要同步的操作类型,完成同步所需要的配置信息,随后给每张表生成 Insert 触发器、Update 触发器、Delete 触发器。在接收端,根据用户的选择初始化镜像数据库,开始增量同步。程序在同步过程中无需用户的参与,自动完成跟踪记录的提取、优化、压缩、网络发送、接收、缓存、写镜像数据库,如果出现暂时的网络中断,程序会报警,并自动重连,仅当程序遇到严重错误,如网络中断 1 小时以上,则需要人工参与排出故障。待故障排除后,只需重新启动同步系统即可。

3.1 跟踪记录表的设计

跟踪记录表主要是用来存储源数据库中对需要同步表的同步操作的信息,信息的主体就是与该操作对应的 SQL 语句。如有表 table_ test(id,name,age),其

中 id 是该表的关键字段,配置了如前所述的三个跟踪触发器,如果对该表添加一行记录(1,'Sun',24),则跟踪记录表中就新增一条 SQL 语句字段为 insert table_test(id,name,age) values(1,'Sun',24)的记录,同样的 Update 和 Delete 操作也将会被记录下来,运行一段时间跟踪记录表的 SQL 语句字段可能有以下信息:

(1) insert table_ test (id, name, age) values (1, 'Sun',24)

(2) update table_test set name = 'SunYeYi', age = 28 where id = 1

(3) insert table_ test (id, name, age) values (2, 'Elvis',21)

(4) update table_test set name = 'SunYeYi', age = 28 where id = 1

(5) update table_test set name = 'Sunyy', age = 28 where id = 1

(6) delete table_test where id = 2

仔细分析上面的六个语句可以看出,如果同时将这六个语句传过去的话,其中只有第 1 条语句和第 5 条语句是有效的,所以这里可以根据操作的表名、操作的类型和操作记录的关键字信息对 SQL 语句进行优化,以去掉冗余的 SQL 语句,文献[5]给出了类似的优化算法。为了方便此优化的实现,跟踪记录表里还要记录下操作的表名、操作的类型和关键字段信息。

因为表里的关键字可能是字段的组合,这给跟踪信息的记录和 SQL 语句的优化带来不小的麻烦,为了统一处理,我们把表的自增标识列作为关键字,如果需要同步的表没有自增标识列,则可以为表添加一列自增标识列(DBS_ID),这对使用源数据库的应用程序是完全透明的。另外再增加一条同步状态字段为同步模块使用。根据以上的讨论,笔者最终的跟踪记录表设计如表 1 所示。

表 1　跟踪记录表

字段名	类型	描述
ID	Int	IDENTITY(1,1) NOT NULL,/*序号*/
SQLString	varchar (2000)	COLLATE Chinese_ PRC_ CI_ AS NOT NULL,/*SQL 语句*/
ExecTime	Datetime	NOT NULL,/*操作时间*/
TalbeName	char (255)	COLLATE Chinese_ PRC_ CI_ AS NOT NULL ,/*表名*/
OperatorType	Tinyint	NOT NULL,/*操作类型,1 - insert,2 - update,3 - delete*/
DBS_ID	Bigint	NOT NULL/*同步表的自增标识列值*/
SyncState	Tinyint	NOT NULL,/*同步状态,1 - 未同步,2 - 已发送,3 - 同步成功*/

3.2　用户配置模块的设计

该模块的主要功能是向用户提供灵活方便的设置接口,来设置如源数据库或者镜像数据库名称、它们各自所在的数据库服务器名、用户名和密码,需要同步的数据表名称序列,对每一个需要同步表的需要同步的操作类型的选择,以及同步周期等系统所必须的配置信息,这些信息笔者采用标准的 Windows 配置文件格式,保存在一个 cofig.ini 文件中,并和同步程序的发送端或接收端放在同一个文件夹下。

3.3　触发器自动生成模块的设计

触发器自动生成模块由发送端在同步系统初始化的时候调用,以后仅当数据库的结构发生变化或者需要修改同步内容时才会调用。该模块为系统的核心模块之一,在获得配置信息后,自动为需要同步的表上需要同步的操作生成相应类型的触发器。如需要同步表 table_test 的 update 操作,则会生成一个表 table_test 的 update 触发器 TrForDBS_table_test_Update,在触发器里需要生成与对该表 update 操作的对应的 SQL 语句,则在编制触发器 TrForDBS_table_test_Update 的时候需要根据表名获取表 table_test 元数据信息,如表所在的数据库名称、表的字段信息,是否有自增标识列等,针对 SQL Server 2000 我们可以从系统表 syscolumns 获取这些信息,然后编程得到触发器的创建脚本,最后将该脚本在源数据库中执行。

生成的触发器 TrForDBS_table_test_Update 的核心语句如下:

```
Insert into DBS_ TRecordsModified ( ExecTime,
SQLString, TalbeName, OperatorType, DBS_ ID)
  Select getdate(),
    'Update ' + DB_Name() + '..table_test set '
    + 'name=' + isnull('''
    + replace(ltrim(rtrim(convert(char(10), ins.
name))),'''',''''''')
    + '''','null') + ','
    + 'age=' + + ltrim(rtrim(str(ins.ID)))
    + 'Where '
    + 'ID=' + ltrim(rtrim(str(ins.ID))),
    'table_test',2,ins.ID
  From inserted as ins
```

该语句完成在应用程序 update 表 table_test 的时候组成对应的 SQL 语句,连带执行时间等其它相关信息写入跟踪记录表中。同样地可以生成 insert 和 delete 触发器。

3.4　镜像数据库初始化模块的设计

该模块分为发送端和接收端两部分,两部分协同工作完成镜像数据库的初始化工作。发送端和接收端的主要步骤如表 2 所示。

表 2 镜像数据库初始化步骤

发送端步骤	接收端步骤
1. 设置数据库故障还原模型为“非简单模式”	1. 接收端程序启动,侦听网络
2. 禁用跟踪触发器	2. 接收文件一
3. 清空跟踪记录表记录	3. 接收文件二
4. 截断数据库日志	4. 获取镜像数据库的排他访问权
5. 完整备份数据库至文件一	5. 从文件一还原数据库(此时数据库处于加载状态)
6. 设立事务标记 EnableTrigger,启用跟踪触发器	6. 从文件二还原日志到指定事务标记 EnableTrigger
7. 备份数据库日志至文件二	7. 删除镜像数据库的所有触发器、约束
8. 关闭数据库连接	8. 设置数据库故障还原模型为“简单模式”
9. 连接接收端	9. 启动同步模块接收端
10. 传输文件一	10. 向发送端返回镜像数据库初始化成功标志(信号 0)
11. 传输文件二	
12. 等待接收端返回镜像数据库初始化成功标志(信号 0)	
13. 启动同步模块发送端	

上述步骤省略了错误处理,如果运行成功的话,对于 1G 数据量的数据库整个过程耗时约八分钟左右,在同步模块启动前,跟踪记录表的记录数约有几千至几万条,由于采用了事务标记还原技术,上述过程保证了初始的镜像数据库加上跟踪记录表的信息与源数据库中的对应信息一致。对此过程的几点说明如下:

(1) 因为网闸的存在,笔者采用了整数来标志状态,如数字 0 表示镜像数据库初始化成功,确保从接收端返回的状态信息不会超过四个字节。

(2) 由于需要做日志备份,所以需要源数据库开启故障还原模型为完全,而镜像数据库在必要的时候可以由源数据库重新初始化,因而镜像数据库的故障还原模型为简单,以提升镜像数据库的性能。

(3) 数据库中的数据一致性、完整性、正确性完全由源数据库来保证,各同步表的信息均各自同步更新,所以必须要删除镜像数据库中的所有触发器、约束。

如果该过程中出现错误,如网络错误、数据库连接错误等,则需要人工排除错误,重新启动该模块。

3.5 同步模块的设计

同步模块是该系统的另一个核心模块,分为发送端和接收端两部分,发送端定时地从源数据库中的跟踪记录表中提取跟踪信息,经 SQL 语句优化模块、压缩模块处理,交给发送模块打包发往接收端,接收端在收到数据包后先进行解包,解压缩,然后在镜像数据库上执行,成功后返回状态信息,发送端在接收到镜像服务器端执行成功的信息后,删除已提取部分的跟踪信息记录,再定时从跟踪记录表中提取新的跟踪信息,如此周而复始完成镜像数据库数据的增量同步。

4 小 结

根据文中的设计思路,笔者主导实施了某电厂的二次系统安全防护项目,整个同步系统对源数据库中的运行几乎没有影响,而且因为将源数据库的查询应用转移到了镜像服务器上,源数据库的性能在某些情况下甚至会比原来有所提升。镜像数据库中的数据时延最低可为 3 秒左右,完全达到了电厂远动数据的应用需要,经受了每分钟上万条记录同步的考验,运行良好。另外本系统配置灵活,模块划分清晰,可扩展性好,如经过对触发器生成模块和镜像数据库初始化模块的简单修改可以使本系统适用 Oracle 等其他数据库。由于应用的实际情况,本系统在设计时并未考虑数据库字段类型为 ntext, binary, varbinary, image 的情况。

参考文献:

[1] Liu Peng, Jajodia S, McCollum C D. Intrusion confinement by isolation in information systems[J]. Journal of computer security, 2000(7):23-28.

[2] 代卫星,章 俊.物理隔离在电力调度专用网中的应用[J].广东电力,2006(7):28-30.

[3] 肖永田,章 军.基于内存交换的网闸系统的研究与实现[J].计算机工程与应用,2005(36):140-142.

[4] 刘 伟,佟俐鹃.异构数据库集成中的变化捕获方案设计[J].计算机应用研究,2005(7):213-215.

[5] 黄晓微,陈 玲,魏 伟,等.基于快照日志分析的数据同步方法[J].后勤工程学院学报,2006(2):59-62.

(上接第 174 页)

[J].系统工程理论与实践,2003(1):49-55.

[5] 方红雨,崔逊学.基于遗传算法的调度问题研究[J].电脑与信息技术,2001(2):1-5.

[6] Hart E, Ross P. A systematic investigation of GA performance on job shop scheduling problems[J]. Lecture Notes in Comp Sci, 2000, 1803:277-286.

[7] Adams J, Balas E, Zawack D. The shifting bottle neck procedure for job shop scheduling[J]. Management Science, 1988, 34(3):391-401.

[8] 玄光男,程润伟.遗传算法与工程设计[M].北京:清华大学出版社,2000.

[9] Cheng R, Gen M, Tsujimura Y. A tutorial survey of job-shop scheduling problems using genetic algorithms-representation [J]. Computers & Industrial Engineering, 1996, 30(4):983-997.

[10] 谢胜利,黄 强,董金祥.求解 JSP 的遗传算法中不可行调度的方案[J].计算机集成制造系统,2002(11):902-906.