银行综合业务系统储蓄子系统的性能优化

汪红莲　（中国建设银行南通市分行　226006）

前言

近年来，随着建设银行业务的迅速发展，综合业务网络系统的数据量不断增加，97 年建设银行总行又提出了二级分行会计、储蓄帐务集中管理的要求，进入系统的网点数日益增多。所有这些都加重了综合业务网络系统的压力，系统运行效率问题逐渐成为能否真正实现二级分行数据集中存放、适应业务不断发展需要的一个关键性的技术问题，本文作者结合两次参加系统优化开发的实践，就如何在不增加硬件投资的情况下，从系统和软件两个方面总结了对储蓄子系统的优化方法，并举例作了详细分析。

一、系统简介

目前江苏省建设银行推广使用的综合业务网络系统为总行推广的湖北版本，早在 1995 年 7 月，省分行就集中技术力量将其中的储蓄子系统移植到美国思群计算机系统公司的 SE20 小型机上，后经过多次的修改完善，基本满足了我行储蓄业务发展的要求，整个系统运行正常。到目前为止，全省已有苏州、无锡、常州、南通等市上了综合业务网络系统，实现了市内的通存通兑，今年常州与南通行率先开通了全省联网系统，实现了省内通存通兑；不久将开通储蓄卡业务，实现与信用卡系统的大联网。

该系统的开发环境如下：

主机支撑软件

操作系统	DYNIX/PTX(R)	4.1.3
INFORMIXONLINE		7.1.4
INFORMIXSQL		6.0 UC1
INFORMIXESQL/C		7.1 UC1

客户机支撑软件

操作系统　AT&T UNIX V 4.2/SCO UNIX V 3.2.4

INFORMIXSE	4.1
INFORMIXSQL	4.1
INFORMIXESQL/C	4.1
标准 C 语言	
网络支撑软件	
程序设计　TCP/IP SOCKET	
广域网	X.25 CCITT 1984
局域网	ETHERNET
点对点	SLIP

二、问题的提出

由于银行业的竞争日趋激烈，为了争取在存款占比上处于领先地位，建设银行抓住改革机遇，不断发展会计、储蓄业务，尤其在近两年，营业网点的业务量迅速增加，处理速度明显下降；此外，根据总行提出的依托城市综合网实现会计（储蓄）帐务集中的战略方向，县级行的数据将全部集中存放于二级分行，这样市分行综合网要容纳的网点数就会几倍、甚至数十倍地增长，尤其是储蓄网点数将会急剧膨胀。储蓄系统中每天必做的平衡检查，要长达几个小时，有时超时不能完成；97 年常州建行在 7 月 1 日晚清理流水帐时，因记录太多，超时，未能清理成功；今年南通建行 6 月 30 日活期结息后，7 月 1 日晚上轧帐竟用了六个多小时，如果全部的县行（包括乡镇）所有数据都加到综合网中，情况会更糟糕。因此，如何对系统进行优化、提高运行效率已成为能否真正实现二级分行数据集中、适应业务不断发展需要的一个关键性的技术问题。

三、优化方法

江苏省建设银行针对系统性能问题于 96 年 9 月、97 年 7 月组织了省内部分计算机骨干力量和公司有关技术人员对综合业务网络系统的储蓄子系统进行了优化，笔者作为主要程序员参加了两次优化工作，经过大家的共同努力，系统性能比以前有了较大提高，结合实际开发过程，总结优化方法如下：

1. 系统方面优化

(1)磁盘空间的分配策略

①在给 ONLINE 分配磁盘空间时，采用原始裸磁盘(RAW DISK)方式，而不采用已加工文件(COOKED FILE)方式，前者比后者处理速度要快得多，且可靠性高；

②逻辑日志及物理日志读写较频繁，应将其放在最快的磁盘设备及磁盘柱面最中心处，不要将逻辑日志和物理日志放到根 DBSPACE 内或与其他活跃表共享同一设备；

③根(ROOT)DBSPACE 要放在磁盘柱面的中心处，一些经常存取的表也应放在磁盘设备靠近中心的位置，这样可缩短读盘存取时间；

④对于随机性的 I/O 应用程序，要将数据分散放到多个磁盘上。用命令 onstat —giof 得到磁盘每秒 I/O 操作的次数，当其值接近 50 时，应设法在磁盘间分离数据，从而降低磁盘每秒 I/O 次数。对于连续的 I/O 通道，应分配给每个 I/O 通道不超过 3 个磁盘。

⑤用 DBSPACETEMP 环境变量将 tmp 空间单设，使数据空间与 tmp 空间在不同的盘上，减少磁头移动时间；同时设置多个存放临时文件的 DBSPACE，这样对并行排序及并行索引的建立会产生良好的效果；用分片(fragment)方法建表时，要将不同的 workdb 放在不同的盘上，并注意考虑互相读的数据要放在临近盘中。

(2)共享内存参数的优化调整

①共享内存缓冲区(BUFFERS)的调整。通常情况下，高速缓冲区读的百分比应大于 95%，写的百分比应大于 82%。有时增加共享内存缓冲区并不能增加高速缓冲区读写百分比，此时用户可考虑调整 PAGE - CLEANING 参数，读参数不合适时，也会引起读写百分比低；

②日志缓冲区大小的调整。物理和逻辑日志缓冲区的最佳大小依赖于用户所处的环境，对无缓冲日志的数据库，PAGES/IO 值接近于 1，对有缓冲日志的数据库，PAGES/IO 值应很接近 BUFSIZE 值。当 PAGES/IO > = BUFSIZE * 0.75 时，说明缓冲区利用率很高，可考虑增大日志缓冲区以减少磁盘的物理 I/O 次数，当 PAGES/IO < BUFSIZE * 0.75 时，说明缓冲区未被充分利用，可考虑减少日志缓冲区，释放共享内存空间；

③页刷新进程(PAGE - CLEANER)的调整。页刷新进程由配置参数 CLEANERS 设置，它的大小由用户硬件决定，最小值为 7，最大值为 32。一般一个独立的物理磁盘设备配置一个页刷新进程，当增加页刷新进程能改善性能时就增加，否则不要随便增加；

④LRUS，LRU - MAX - DIRTY 及 LRU - MIN - DIRTY 的调整。LRUS 参数指定系统中使用的 LRU 队列数，LRUS 的缺省值为 USERTHREADS/2 和 8 的最小值，USERTHREADS 配置参数指最大用户线程数。LRUS 的最小值为 3，最大值为 32。LRUS 的最佳值取决于用户的硬件配置，选取它的最好方法是用不同的值去试验，同时监测相应的性能。对多 CPU 的机器，较大的 LRUS 值能提高其性能。LRU - MAX - DIRTY 和 LRU - MIN - DIRTY 分别为一个 LRU 队列中允许的非空闲缓冲 所占的百分比的上限和下限。当非空闲缓冲区所占的百分比超过 LRU - MAX - DIRTY 时，页刷新进程将数据由缓冲区写到磁盘，当该百分比低于 LRU - MIN - DIRTY 时，页刷新进程将停止由缓冲区向磁盘写数据。LRU - MAX - DIRTY 的最佳取值范围为 40 - 70，LRU - MIN - DIRTY 为 30 - 60，必须保证 LRU - MIN - DIRTY 比 LRU - MAX - DIRTY 小。

2.软件方面的优化

(1)突破原有的联机事务处理(OLTP)方式，直接使用 Informix 嵌 C 语句，少用 C 语言程序，多用 Informix 语句，如采用存储过程、触发器等 Online 并行数据库系统本身提供的多线索(MultiThreads)机制。

(2)在批处理程序(如平衡检查)中，设定 OPTCOMPIND 环境变量值为 2——考虑成本优化方式，设置程序执行的并行度 pdqpriority，最大可取 100。

(3)在程序中尽量减少使用游标，如按机构、按储种循环等。经比较发现使用游标效率最低。

(4)尽量避免使用游标的嵌套，改用 SELECT 语句将多表实现连接，从数据库中一次性取出需要的那部分数据。

(5)对一些常用的表，如机构编码表等，可建临时表，使用多少字段即取多少字段，减少冗余，提高速度。

(6)在进行多表连接时，互相连接的表中，必须至少有一张表不带索引，这样可保证进行杂凑连接(HASH - JOINING)，实践证明，这种方式速度较快。

(7)某些类似单用户方式的批处理工作(如清除过期数据，活期结息)等，可考虑在先做备份的情况下，去掉逻辑日志，执行本功能，执行过程中没有任何问题，则结束后恢复逻辑日志，若发生异常，可恢复数据再做。

(8)在做某些批量清理工作(如清理过期数据、清理流水)时,如果某张表记录较多时,"删除记录"的效率远远不如"卸出有用数据→删表→再建表→装入有用数据"的效率高。

(9)由于磁盘上数据的插入删除是不断进行随机变化的,很难保持初始的生成状态,为此尽量使数据在物理上和逻辑上保持一致性,定期卸载数据,定期清理无效数据,定期建聚类索引。

(10)对查询程序,尽量多使用精确条件,少使用模糊条件。将"matches"、"like"等模糊条件语句改为"="之类的精确匹配方式,因为像 matches 等在 INFORMIX 中效率较低。

(11)使用系统优化编译"CC —o"也可以提高部分效率。

四、优化示例与分析

1. 分片建表

对记录数很多的表,可考虑将其分片建表。假设现有三个数据库空间 workdb1、workdb2、workdb3,对定期帐户表 satdq 分片建表如下:

(1) create table satdq fragment by round robin in workdb1, workdb2, workdb3;

(2) create unique index satdq-idx on satdq(zh)
fragment by expression
zh between "32064525101000000000" and "…" in workdb1,
zh between "…" and "…" in workdb2,
reminder in workdb3;

(3) alter table satdq add rowids;

其中第一句表示用 round robin 方式建表,第二句表示用表达式方式建索引,第三句是附加的,是为了该表建 forms 时能正确查询。

分析:将一张表中的数据放在不同的磁盘上,当对定期户主表(70 多万条记录)进行存取操作时,多个磁盘上的磁头同时在进行数据的读出与写入,提高了并行处理的效率,速度有一定的提高。如果大量数据放在同一张盘上,对定期户主表进行存取操作时,磁头要在一个磁盘上来回频繁移动,机械的速度制约着数据处理的效率。其他的大数据量的表如各种明细帐也可采用类似的分片方法建表,以提高整个储蓄子系统的效率。

2. 清理流水

储蓄系统湖北版中,清理流水的流程如下:

对清理流水,当记录多时,"delete"效率不如"drop table"效率高,将"delete"改为:

unload to temp select … from table→drop table→create table→load from temp insert into table

分析:以 10 万条交易流水为例,假设从轧帐时间后有 5000 条流水。

第一种作法开销:

锁:(100000 − 5000) * 2 = 190000

逻辑日志:190000 删除操作

物理日志:95000 * 2 * 每条记录的字节数

第二种作法开销:

锁:unload to temp select … from table 5000 * 2 = 10000 +

load from temp insert into table 5000 * 2 = 10000

共计 20000,可忽略不记。

逻辑日志:可忽略不记

物理日志:5000 * 4 * 每个记录的字节数

并且 drop table 和 create table 不会因记录数增多而改变速度或增加消耗。

3. 平衡检查

现例举储蓄子系统的数据平衡检查中的"总帐与分户帐的一致性检查"功能。

功能目标:按机构、按储种统计出各分户帐的户数和余额,与总帐数据进行核对,如有不一致的情况,则将出错结果写入一文本,否则将"完全正确"写入文本。

程序略。

分析说明:(新程序)

①putenv("OPTCOMPIND = 2"),是设定 OPTCOMPIND 环境变量的值为 2——考虑成本优化方式,在这种设置下,系统会从最佳执行效果的角度来优化我们所写的 SQL 语句。

②set pdqpriority 100;设定本段程序执行的并行度为 100%。

因为储蓄子系统是在轧完帐,清理流水后,做平衡

检查工作，此时，一般是单用户状态执行，没有其他的OLTP或后台处理，设定最大的并行度有利于充分利用CPU资源，有效提高运行速度。

③通过多表连接，再利用SQL本身提供的聚合函数sum和count进行统计，方便快捷，屏弃了原来使用多重游标、循环嵌套，效率较低的做法。

改进以后，摈弃了原来用游标逐个累加、比较的方式，效率明显提高。整个程序简洁、清晰。如，南通行的储蓄数据执行总帐与分户帐一致性检查，用新程序只花了10分钟时间，而老程序则需要一个多小时，大大缩短了执行时间，运行速度明显提高。

五、存在的问题及改进措施

1.系统方面

(1)逻辑日志增长过快。南通行在七月一日轧活期结息的帐(大约有8万笔流水帐)时，由于执行时间较长，逻辑日志增长过快，长事务来不及滚回，导致INFORMIX数据库系统崩溃，轧帐无法成功完成。

改进：①需调整INFORMIX—ONLINE的pdq捆绑参数，原为10，现改为NO。

②逻辑日志缓冲区大小由原来的1024K改为256K。

(2)公用的PRINT目录下文件数不够。由于湖北的综合业务网络系统，无论是储蓄、会计、还是信用卡子系统，都是将生成的打印文件、前台查询形成的文件、每天需下传到各网点的文件等各种临时文件放于$HOME/print目录下，所以过一段时间，PRINT目录下的文件就会很多，应用系统运行时，有时不注意会造成文件"写失败"，严重时会造成网点无法正常营业。

改进：①系统管理员定期清理PRINT目录下的文件。如，每周一次或每三天一次，视业务量的大小定。

②将每天PRINT目录下形成的文件移到一个其他目录下，如，PRINT1、PRINT2……PRINT7下，然后每周清理这些目录。

③要求程序员修改程序，做到"随做随清"，下传成功后，就将PRINT目录下的文件清理掉。

2.软件方面

(1)按储蓄日终轧帐目前的流程，根据流水写分录的过程中，需要从流水表中读一笔流水，再根据流水中交易码查找分录规则，然后按规则生成对应分录，再将分录写入轧帐表(satzzb)中，通常情况，一笔流水生成至少两笔分录，最多六笔分录，也就是在拆分录过程中有两次读硬盘操作，两到六次写盘操作。而在后面的操作中还要再对轧帐表(satzzb)进行读操作，这样会导致磁盘I/O较忙，为减少磁盘I/O，可以把轧帐表写到内存中，在内存中对其进行读写操作，以提高效率；审查平衡和汇总所使用分录直接从内存中取出，可加快速度。但由于数据量大，对系统有如下要求(以每天10万笔交易估计)：

每笔交易平均产生2～6笔分录，以4笔为平均数，每日平均产生40万条分录记录，每条记录80bytes，所需共享内存为：40,0000 * 80 = 32 M bytes

另外，对共享内存中访问记录需设计较好的算法。

(2)数据平衡检查中"储蓄分户帐与明细帐一致性检查"功能，因明细表数据太多，所以效率很低。

本人建议：

①只检查每日动户数据；

②在生成每一笔明细时，按满页笔数预分空间，使每一帐户明细记录在物理上连续，减少磁头移动；

③在生成明细时，即进行明细连续性检查，并且只检查最后一笔明细与本笔的连续性；

④定期(如按月)将明细表中数据过渡到另一张同结构的表(如过期明细表)中去。

六、结束语

本文只是针对建设银行湖北版综合业务网络系统结合开发实践，提出了银行系统性能优化的几种方法，希望能对各位同行有所裨益，起到抛砖引玉的作用。本人认为应将性能优化工作作为一个系统工程来整体考虑，只有〈系统〉+〈软件〉的不断优化调谐，才能有效地缩短机器运行时间，提高整个系统效率，充分发挥现有机器的潜能；从另一方面讲，也可在减少硬件投资的情况下处理更多的金融业务，为建设银行节省一笔重复投资的资金，发挥出较大的经济效益。

(来稿时间：1998 年 8 月)