基于 C/S 结构应用系统性能优化方法的探讨

关宇平 （广州石油化工总厂信息中心 510726）

摘要：本文主要讨论从应用程序性能和数据库性能两方面对一个基于 CLIENT/SERVER 结构的应用系统进行优化的方法。

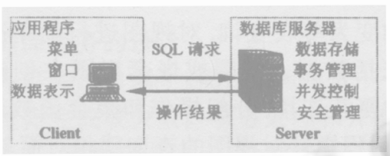
关键词：CLIENT/SERVER 结构 数据库管理系统 DBMS 性能优化 磁盘输入/输出(磁盘 I/O) 网络 I/O ORACLE7 PowerBuilder(PB)

一、问题的提出

客户机/服务器(Client/Server 以下简称 C/S)计算技术是企业近年来开发信息管理系统所使用的主要技术。但在应用系统开发完毕，投入使用后，开发人员往往发现：随着系统的用户数和系统包含的数据量巨增，系统的性能产生了很大波动，主要表现在系统的响应速度不尽人意，因此，如何优化基于 C/S 结构的应用系统，提高系统运行效率是开发人员亟待解决的问题。

二、C/S 应用系统性能指标及优化原则

一个基于 C/S 技术的系统由三个部件组成：数据库服务器(Server)，客户机(Client)，以及服务器与客户之间的通信网络。其工作原理如图所示：



(1)评价 C/S 应用系统性能的指标是：

·系统响应用户查询的时间

·数据库服务器的吞吐量(每秒的事务处理量)和并行处理能力

(2)由原理图可知：C/S 系统中，多个客户机通过网络来访问 SERVER，因此 C/S 系统优化的原则是：

·尽量减少系统的瓶颈：网络 I/O 和网络的信息(数据)传输量。

·在 Client 与 Server 之间合理地分布应用系统的功能，使两端的工作负载达到平衡。

(3) 性能良好的 C/S 应用系统的硬件基础是：支持较宽带宽的网络；有快速 CPU，快速硬盘，大容量内存，支持并行多处理的数据库服务器；快速 CPU，足够内存的客户机。因此，一个 C/S 应用系统的优化任务应由网络管理员，系统管理员，数据库管理员(DBA)，和应用程序开发人员合作完成，本文主要讨论做为一名 DBA 或开发人员如何对 C/S 系统性能进行优化的问题，由于 C/S 系统的应用分布在 Client 端和 Server 端，其性能也被分为应用程序性能和数据库性能两方面，本文将根据笔者的实践经验，从上述两方面分别探讨 C/S 系统性能优化的方法。

三、数据库性能的优化

1. 设计良好的应用数据库是保证系统性能的基础

数据库是存储与管理信息最有效的方式，是一个应用系统的核心，系统成功与否关键是数据库的设计。为提高系统的性能，数据库设计应遵循下列原则：

(1) 基础表(如代码表)的数量及属性越少越好，尽量简化。

(2) 数据表主键的定义应符合三个特性：唯一性；静态性(主键被赋值后，应当在该行的整个生命周期中保持不变)；简洁性(主键作为表间连接的工具，其包含的列越少，表间的连接就越简单)。

(3) 数据表列数据类型的设定不仅要符合实际业务需求，还要考虑对数据库服务器存储空间的利用率和对系统性能的影响。例如字符型数据数据类型的使用：当开发者可在固定长度和可变长字符类型进行选择时，ORACLE7 建议开发者使用变长 VARCHAR2 数据类型，而不是定长 CHAR 型，特别是对大型表，因为使用 VARCHAR2 型可节省存储空间，而且 ORACLE 在需要对一个包含 VARCHAR2 列的大表进行全表扫描时所读的数据块显然少于将相同的数据存放于一个 CHAR 列中所读的数据块。

(4) 认真考虑非规范化与规范化表之间的折衷。遵循规范化理论进行数据表设计可减少数据冗余,提高了联机事务处理(OLTP)的效率,而非规范化则面向数据查询,通过增加冗余数据来减少表间连接,降低了数据库服务器 CPU 的处理负载和磁盘 I/O 操作,提高了联机分析处理(OLAP)的效率,但非规范化会增加存储空间的消耗和为保持冗余数据的一致性所带来的额外系统开销。开发者应根据应用系统的侧重点:OLTP 还是 OLAP 来确定表的规范化程度。(5) 在 SERVER 上创建数据表时,应尽量加入数据完整性和业务规则的约束。

如在表上定义主键,外键,唯一性约束,有效性检查,值域范围约束等,这样可降低应用程序编码复杂性,提高维护程序的效率,保持数据一致性,能真正意义地拒绝非法数据进入数据库,而在应用程序端定义数据约束条件也是有必要的,这可使用户录入的数据在提交给数据库之前先进行有效性校验,防止无效的数据在网络上传输。

2. 服务器端编程的目的:减少网络传输量,提高系统响应速度

一个能担当 C/S 系统中数据库服务器角色的 DBMS,除负责定义,维护,存取数据外,通常还支持触发器,存储过程等用于减少网络 I/O 的特性。

(1) 数据库触发器(Trigger)是特殊类型的存储过程,可用于实施复杂的业务规则,其优点是集中进行规则实施,自动执行,避免不必要的网络 I/O。

(2) 存储过程(Procedure)存放在数据库,是一组被编译的 SQL 语句,DBMS 已对该过程进行了语法检查,预分析,并对其中的 SQL 语句建立了查询计划和优化,而从 Client 向 Server 发送的 SQL 语句是未编译的,在运行之前均要经历分析优化。在 Client 调用存储过程,通过网络发送的数据只有调用命令及参数值,因此,存储过程不仅为在数据库上处理 SQL 事务提供了一种最快途径,还大大降低了网络传输量。在应用中,如有复杂的需要涉及大量表连接和子查询操作的 SQL 语句或在循环过程中需反复调用的 SQL 语句,应将这些语句创建为存储过程。

在系统开发阶段,由于数据库的建设还不完善,业务规则未定型,开发人员在遇到无法用数据完整性表示的业务规则时,常规做法是在应用程序中去实现,在系统投用后,用户对系统的业务功能认可后,可把部分在应用程序端实施的复杂业务转换成触发器或存储过程,移植到服务器上,可大幅度减少从应用程序端存取数据库的复杂性,提高系统的效率。

3. 恰当得使用索引以提高查询速度

索引避免了数据全表扫描,加快了记录检索速度,降低了磁盘 I/O,但索引需要额外的数据库存储空间,并会在数据库操作期间增加一些额外的开销(数据表的维护需同时维护该表所带的索引),因此对一个数据表不能创建太多的索引,通常在查询出现瓶颈处对查询子句中经常使用的字段或经常用于表连接的字段创建索引。由于不同的 DBMS 对索引的处理有不同的机制,对索引的使用也有不同要求,开发人员要根据实际情况恰当选择和创建索引。

4. 优化 SQL 查询语句

SQL 查询语句是应用程序的重要部分,质量不高的 SQL 语句是应用程序产生性能瓶颈的主要原因,象 ORACLE 或 SQL SERVER 等大型关系型数据库,缺省使用的是基于成本(COST BASED)的优化器,即对一条 SQL 语句,优化器按各种可能的访问路径生成一系列的执行计划,并通过计算(或估计)选择一个使用数据库系统资源最少的计划来执行 SQL 语句。开发人员应在理解所选用数据库的优化器工作模式的基础上编写 SQL 语句,SQL 语句编写的关键是条件查询(where)子句的编写,原则是使优化器能有效利用数据表上创建的索引,或是尽量缩短优化器的优化步骤,使优化器能快速产生效率最高的执行计划。

5. 应付高代价查询的数据库策略

所谓高代价查询就是需进行多表连接,扫描大量的数据才能得到希望的结果,如在决策支持系统(DSS)或执行信息系统(EIS)中针对海量业务数据的大型综合性报表和分析图表的生成等。这类查询操作会极大增加数据库服务器的负担,耗费大量资源,应付策略有:

(1) 使用数据视图(VIEW)屏蔽复杂的 SQL 命令,使多表查询变为单表查询,对视图进行简单查询就可获得复杂的概括数据。视图或嵌套视图在创建时,DBMS 对其包含的复杂连接,分组操作和集合函数进行了编码和排错,应用程序针对视图的查询就相对简单和高效。

(2) 将复杂逻辑计算和查询生成的结果以报表的形式存储,用户再次以相同条件查询只打开相应的报表文件。如利用 PB 中的数据窗口(DW)可将查询结果生成报表 PSR,文本 TXT,EXCEL 等格式的文件。

(3) 引入数据仓库的概念和方法:根据特定的分析内容建立主题数据表,在应用程序中周期性扫描业务数据,通过对数据的抽取,转换,综合统计来定时更新主题数据表的内容,主题数据表极大提高了针对特定分析的效率。

6. 调整数据库服务器的性能

这是调整 C/S 系统性能的难点,需要数据库管理员 DBA 针对相应的 DBMS 进行特别的调整。主要调整项

有：

(1) 整理数据表所在物理磁盘的碎片，调整 SERVER 内存中的高速缓存，以减少磁盘 I/O。

(2) 分散数据，消除服务器中因访问文件或数据而引起的磁盘 I/O 竞争。

在 ORACLE7 中，数据文件与事务日志文件应分散存储在不同的物理磁盘，使事务处理执行的磁盘访问不妨碍对相应事务日志进行登记的磁盘访问；数据文件与相应的索引文件应分散存储，使事务执行时可以并行读表和索引数据，消除磁盘竞争。

(3) 从数据库对象(数据表，索引等)的低层存储方面考虑如何减少数据碎片并提高磁盘 I/O 性能。ORACLE7 为数据库对象提供了不同的存储参数(如 PCTUSED, PCTFREE 等)，正确设置存储参数可充分利用磁盘空间并发挥磁盘 I/O 的最佳性能。

(4) 确定服务器是否支持并行多处理，在 ORACLE7 中启用并正确配置并行查询选件能极大优化服务器的查询性能。

四、应用程序性能的优化

应用程序的性能表现在：SQL 请求发生后，返回结果的显示速度，含有数据的屏幕之间的切换速度；含有复杂业务逻辑处理屏幕的运算速度等方面。

应用程序性能调整的目的是使用户满意程序处理业务的速度，同时又能尽量减轻数据库服务器的负担，减少因返回值引起的网络传输量。

调整应用程序性能的策略(以开发工具 PB 为实例说明)：

1. 数据库事务的设计

数据库事务是由多个 SQL 语句组成一个逻辑单元，作为一个整体完成或失败。事务处理和并发控制由数据库服务器执行，而事务起点与终点的划分则由应用程序实现，应使数据库的事务尽可能短，短的事务使事务中由 SQL 语句获得的锁和占用的回滚(ROLLBACK)空间尽快释放，从而减少了并发处理时系统资源的竞争。

2. 限制由数据库服务器返回的信息量

当用户对数据量很大的表进行自定义条件查询时，应设定条件防止他们因不经意的选择而返回太多的数据，因为：(1)大量数据的查询会给数据库服务器造成不必要的负担，降低 DBMS 的性能；(2)大量的返回数据引起网络通信量大增，引起不必要的拥挤；(3)如果返回数据量超过客户机的可用内存会引起客户机发生严重错误。

3. 尽量减少对数据库服务器的访问次数，以减少网络 I/O

如应用程序需要反复操作相同或类似的数据(如代码表)，在 PB 中，可利用数据共享技术，数据一次从数据库检索之后，存储于客户机缓存中，被反复使用；也可利用数据窗口事先存储基本固定不变的数据，使用时不必去服务器检索等方法来提高应用程序的效率。

4. 加速屏幕切换或数据显示速度

(1) 不要在一个窗口(屏幕)放入太多的功能，过于复杂的屏幕执行速度慢。

(2) 在窗口显示之后，再从服务器检索需要的数据，不要让用户对着空屏等待。

在 PB 中的具体实现方法是定义一个用户自定义 ue-postopen 事件，在 ue-postopen 事件中编写数据检索语句，然后在 window 的 open 事件中，添加脚本 script：Post Event ue-postopen()。

(3) 在 PB 中利用数据窗口 DW 检索大量数据时，应选用 DW 的 Retrieve as Needed 选项，这可限制每次从数据库返回到缓冲区的记录个数，效果是用户看到的数据窗口很快填满数据，并得到屏幕控制权，而不是等待所有的数据检索出来后才去填充数据窗口。

五、结束语

一个 C/S 应用系统的目的是为了提高用户的业务处理效率。开发人员不仅要使应用程序满足用户的业务需求，更重要的是保证系统有良好的性能。本文只阐述了性能优化的一些基本规则。系统的优化是个循序渐进的过程，步骤是：发现系统的瓶颈→寻找优化策略→实施优化。往往，系统的一项优化措施可能会引起一个新的瓶颈的出现，这需要开发人员在充分了解前端开发工具和数据库管理系统的前提下，根据应用系统的业务需要(重点是 OLAP 还是 OLTP)对优化措施进行选择和折衷，并与 DBA，系统管理员，网络管理员进行合作，力图找出系统最优性能的平衡点。

参考文献

[1] Singh, Leigh 等著《Oracle 数据库开发指南》清华大学出版社。

[2] 《ORACLE7 Server Application Develop's Guide》ORACLE 公司出版

[3] 何军，刘红岩，王榕等编著《PowerBuilder5.0 原理与应用开发指南》电子工业出版社。

(来稿时间：1999 年 1 月)