提高数据库事务处理性能的中间件设计

王建华　何盈捷　张　孝　杜小勇
（中国人民大学信息学院　北京100872）

摘　要　数据库服务器对于客户端请求都有一个最大的连接数，这限制了数据库事务处理性能，尤其在对一些具有终端操作的事务处理类型的应用，每分钟能处理的事务数不高，表现为TPC-C测试中的tpmC的性能值偏低。本文提出了提高数据库TPC-C测试的tpmC值的两种中间件的策略，在传统的连接池技术的基础上，提出了降低调度粒度，以事务乃至语句——而不是客户端请求——为调度单位的中间件模型，从而可提高数据库的事务处理效率。

关键词　数据库，事务处理性能，TPC-C，中间件

The Design of Middleware to Advance the Transaction Processing Performances of the Databases

WANG Jian-Hua　HE Ying-Jie　ZHANG Xiao　DU Xiao-Yong
(College of Information, Renmin University of China, Beijing 100872)

Abstract　Each database server sets the max value of the connection number to the users, which restricts the transaction-processing performance of the database server, so that the transaction-processing number by second of these systems is not high, and the value of tpmC in TPC-C test is low. For improving the value of tpmC, this paper is based on the technique of the connection pool, and puts forward two middleware models, which scheduling unit is a transaction or a SQL sentence of client's requests to advance the transaction-processing performances of the databases.

Keywords　Database, Transaction-processing performance, TPC-C, Middleware

1　引言

TPC-C是一个工业标准的基准测试，它由事务处理协会Transaction Processing Council (TPC)提出，用来评价数据库在线事务处理(OLTP)的效率。TPC-C模拟终端操作执行数据库的处理，操作表现为订单输入环境，包括处理新订单、付款、订单状态、发货和库存水平。主要的TPC-C测试项目是tpmC (tpm是transactions per minute的简称；C指TPC中的C基准程序)。tpmC的定义是每分钟内系统处理的新订单个数。

每个数据库服务器都设有客户端请求的最大连接数，这限制了数据库事务处理性能，尤其在对一些具有终端操作的事务处理类型的应用，每分钟能处理的事务数不高，表现为TPC-C测试中的tpmC的性能值偏低。提高系统tpmC值的办法有许多，提高每一个连接的使用率，使每一个连接可以同时为多个客户端服务是其中之一。使用中间件技术可以达到这一目的。目前的中间件技术侧重点各有不同，但都是以客户端请求为调度单位，连接的使用效率不高。我们提出的中间件模型，侧重于减少连接的开销，降低连接的调度粒度，提高连接的使用效率，提高每个客户端的响应时间，最终增加数据库可以同时连接的客户端请求数。

使用连接池技术，可以节省建立和释放连接的开销，这些连接始终在内存中，可以复用。但这并没有改变数据库的连接数的限制，若用户没有释放连接，即使连接空闲，其他请求仍不能使用该连接资源。因此，在这种情况下，连接成为一种紧缺的系统资源，需要对此进行精细的调度。为此，中间件需要对客户端的请求进行分析，分解出其中的事务或者SQL语句，然后以比客户端请求更细的粒度(事务/语句)对连接进行申请和释放，从而使一个连接可以为多个客户端请求服务，提高系统的tpmC值。

2　中间件

中间件是介于应用软件和系统软件之间的一类软件，目前，它并没有很严格的定义，但是普遍接受IDC的定义：中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源，中间件位于客户机服务器的操作系统之上，管理计算资源和网络通信。从这个意义上可以用一个等式来表示中间件：中间件＝平台＋通信，这也就限定了只有用于分布式系统中才能叫中间件，同时也把它与支撑软件和实用软件区分开来[6]。

对于客户端来说，中间件封装了各种复杂的技术细节和逻辑问题，简化了使用和开发；对于服务器来说，中间件实现了异构连接和分布式应用，降低了维护和管理的开销，提升了性能。目前流行的中间件有以下几种：

数据库中间件　更加着重于某个特定数据库的某个固定的数据类型的转换[4]；

远程过程调用中间件　适用于局域网，而不适合分布式环境，容错性差；

面向消息中间件　可以保证消息达到目的地时效果最好，适合于恶劣的客户/服务器环境，但不兼容；

面向对象中间件　适合于对象环境中，还没有在大型系统中使用过；

事务处理中间件　提出了事务处理的语法和API，让用户编写分布式应用，实现速度和可靠性，但是比较复杂，不便于使用；

专用中间件　专门为特定的工具或者运行环境优化定

王建华　硕士研究生，研究方向是数据库、智能信息检索。何盈捷　博士研究生，研究方向是数据库、智能信息检索。张　孝　讲师，博士，研究方向是数据库、数据仓库。杜小勇　教授，博士生导师，主要研究方向包括数据库、数据仓库与商务智能、智能信息检索。

制。

上述各种中间件侧重点不同，但都是以客户端的请求为单位进行调度的。服务器可以同时服务的客户端数目，受到可用连接数的限制。它们都没有对客户端的请求加以分析，分解客户端的请求，以更小的单位进行调度。

本文提出的中间件模型，在传统连接池技术的基础上，通过改变调度粒度，分解客户端请求，以事务或者语句为单位进行调度，从而提高每个连接资源的使用效率，减少客户端的等待时间，使数据库可以同时服务的客户端数增加，从而提高数据库的 tpmC 性能。我们称这种中间件为数据库连接中间件。

3 连接池

每当客户端访问数据库时，它必须连接到该数据库上。数据库连接会耗费系统开销，这需要客户端来创建连接和维护连接，并且当不再需要连接时释放它。基于 Web 的应用程序所需的系统开销特别高，因为 Web 用户的连接或断开更为频繁。所以，连接在这里可以看作是一种系统资源[1]。

为了有效地管理连接这种资源，数据库连接池的技术应运而生。连接池最基本的思想就是预先建立一些连接放置于内存对象中以备使用。如图1所示，当程序中需要建立数据库连接时，只须从内存中取一个来用而不用新建。同样，使用完毕后，只需放回内存即可，释放的开销也节省了。而连接的建立、断开都由连接池自身来管理。同时，我们还可以通过设置连接池的参数来控制连接池中的连接数、每个连接的最大使用次数等等，从而提高系统的效率[2,3]。

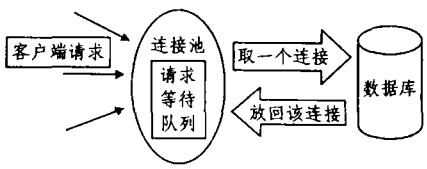

图1

连接池的技术降低了建立和释放连接的开销，充分利用每个连接为客户端请求服务，提高了数据库连接的利用效率。但是，它并没有从本质上改变连接的管理和维护方法，调度单位仍是每个客户端的请求。连接数的限制，从本质上决定了服务器可以同时处理的客户端应用数，决定了每分钟可以处理的事务数。只有改变客户端和数据库服务器之间的连接方法，改变这种调度的策略，降低调度粒度，分解每个客户端请求，以更小的粒度为调度单位，才能进一步改变数据库系统事务处理的效率，提高 tpmC 的值。

4 以事务为调度单位的中间件模型

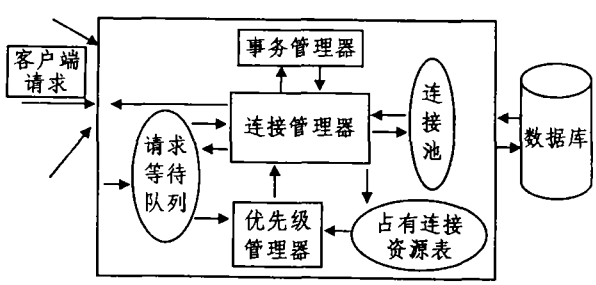

图2

在连接池技术中，每个客户端仍单独占用一个连接，一旦一个连接被一个客户端请求占用，连接池就无法跟踪连接的使用情况，连接的释放由客户端决定。只有客户端退出，连接才被放回到连接池。若客户端没有放回连接，即使目前在进行一些客户端的处理，这个连接空闲，仍不能被等待的其它客户端请求使用。

为了充分利用每个连接，我们提出以事务为调度单位的中间件技术。中间件对每一个客户端请求加以分析，把请求分解成为独立的事务，以事务为调度单位，使每个事务独占一个物理连接，降低了连接使用的粒度，从而进一步提高了连接的使用频率，增加数据库可以同时服务的客户端请求数，提高了数据库的 tpmC 的值。

数据库连接中间件体系结构如图2所示，它的输入是客户端请求，这些请求都是由一系列事务组成。事务管理器把每个客户端请求逐一分解为事务，以事务作为调度单位，申请连接资源，事务逐个被执行并返回执行结果。数据库连接中间件包括以下成分：

请求等待队列 对每个到达的客户端请求，若等待队列中有空闲位置，则加入队列。

占有连接资源表 记录占有连接资源的客户端请求的列表。每一个请求若占有连接资源，则连接管理器就把该请求的 clientID 发送到占有连接资源表；在这表中的请求，不能参加优先级排列，不能再次申请连接；若连接释放，则从占有连接资源表中释放对应的 clientID。

优先级管理器 对等待队列中的请求加以排队，进行优先级排序，返回连接管理器一个优先级最高的请求在等待队列中的 ID。为了保证客户端请求中的事务依次执行，需要请求中的事务施行完毕后，它的下一个事务才能申请连接资源，因此，优先级的排序只能在占有连接资源表以外的请求中进行。为了提高用户的响应时间，所以已经执行了某些事务的请求，优先级总是低于新的请求。

事务管理器 界定每个请求中的事务。对于连接管理器指定的请求，界定它目前包含的第一个事务，把该事务的结束位置传给连接管理器。

连接管理器 中间件的核心模块，监控连接池的连接资源，若有连接空闲，则产生一个新的事务，并维护占有连接资源表，占有连接资源。连接管理器的执行逻辑如图3所示。

```
id=优先级管理器(请求队列);//得到请求等待队列中优先级最高的请求
int clientID=事务管理器(请求队列中的第 id 个请求);//得到该请求目前的第一个事务
if(clientID<0)//该请求的 clientID 为空，表明该请求的第一个事务尚未执行
{
   clientID=新的唯一的 clientID;
   把 clientID 发送给占有连接资源表;
   把新的事务和它的 clientID 传给连接他;//占有连接资源
   if (该请求还有内容)
     把 clientID 发给该请求;
   else //该请求为空
     释放它在请求等待队列中的资源;
   end if;
}
```

图3 以事务为调度单位的连接管理器调度算法

当数据库服务器返回结果时，连接管理器解析出结果中的 clientID，并把结果返回给相应的客户端连接；同时连接管理器释放它对应的 clientID 占有的占有连接资源表中的资源，把连接资源放回连接池。

例1 中间件请求等待队列中可以包含10个请求，每一个请求的事务序列如图4所示，假定数据库的可用连接数为2，调度策略使用最简单的 FIFO。

等待队列中的10个最初的请求如图，按照 FIFO 的策略，t1是等待队列中第一个位置的请求中的第一个事务，t1首先占有连接资源1，赋予它的 clientID 值为1，然后把 clientID 返

回给该请求的剩余事务，它的下一个事务 t2带有该 clientID 值申请资源时，不需要生成新的 clientID；同理，t3是等待队列中第二个位置的请求的第一个事务，t3占有第二个连接资源，赋值 clientID 为2，如图4 i 所示；

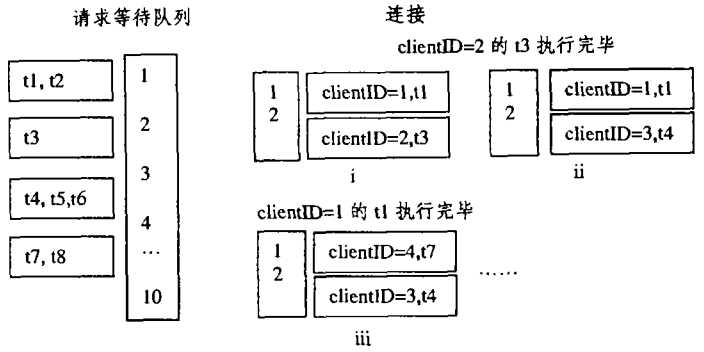

图4

第二个连接资源放回连接池后，由于第一个连接资源上 t1尚未结束，所以该请求的第二个事务 t2不参与竞争连接资源；t4是等待队列中第三个位置的请求的第一个事务，t4占有第二个连接资源，clientID 赋值为3，如图4 ii 所示；

t7是等待队列中的第四个位置的请求的第一个事务。当第一个连接资源放回连接池后，为了避免请求饿死，t7占有第一个连接资源，赋值 clientID 为4，如图4 iii 所示。

以事务为调度单位的数据库连接中间件，适用于短事务，若客户端的长事务过多，每个事务占用连接的时间仍然过长，这时需要进一步降低调度粒度。下一节我们提出以语句为调度单位的数据库连接中间件模型。

5 以语句为调度单位的中间件模型

事务是数据库管理的单位，具有 ACID 特性。若把事务的每一个语句作为调度对象申请连接资源，则在执行语句时服务器必须能够识别语句所属的事务体。因此，以语句为调度单位的中间件模型并不适用于所有的数据库，只有满足以下特性的数据库才可以使用这种中间件模型：每一个连接的开始语句都生成一个新的事务，一个连接包含一个或者多个事务；每个事务都有全局唯一的 transactionID，并根据 transactionID 进行恢复。这种中间件模型把每个事务分成单个语句在多个连接上分别执行，所以每一个语句必须带有所属事务的 transactionID，只要可以识别语句所属的事务，就可以在服务器端还原事务体，服务器仍然能够保持事务的 ACID 特性。

以语句为调度的中间件的结构与图2相同。只有连接管理器和优先级管理器的功能更复杂：工作单位都为语句。

·占有连接资源表　记录的不是 clientID，而是中间件为每一个事务生成的唯一的事务号 transactionID

·优先级管理器　涉及到对已经执行了某些语句的事务、新的事务、新的请求三种情况的优先级判定。

·连接管理器　占有连接资源的不是一个事务，而是一个语句。连接管理器负责监控连接池中的连接资源的使用情况，当有连接资源空闲时，连接管理器需要生成一个语句，占有空闲的连接。连接管理器的执行逻辑如图5所示：

```
id＝优先级管理器(请求队列)；//得到一个优先级最高的请求在队列中的 ID
if(该请求的 transactionID is 空)
{
    调用事务管理器，找到事务结束的语句；//是一个新事务的开始语句
    生成一个全局唯一的 transactionID;
    把该 transactionID 发送到占有连接资源表；
};
if(事务 is not 结束)
{
    把 transactionID 发送回请求队列中的请求；
    在事务体的语句中都标注上 transactionID;
}
把该语句和 transactionID 发送给连接器；//占有连接资源
```

图5　以语句为调度单位的连接管理器调度算法

连接管理器对于服务器返回的结果，首先解析出结果中的 transactionID，由此映射为对应的客户端，把结果返回客户端连接；然后连接管理器释放 transactionID 占有的占有连接资源表中的资源，把连接资源放回连接池。

当中间件把一个语句和它对应的 transactionID 传给服务器后，数据库服务器需要首先确定该语句所属事务，响应的算法见图6。

```
解析出 transactionID;
if (is 新的 transactionID)
  在服务器端的事务池中新建一个事务，生成一个内部的唯一的
      TransactionID;
else //事务池中已经有了这个 transactionID
  将这个语句与这个事务关联
end if;
```

图6　数据库服务器计算语句所属事务的算法

例2　中间件请求等待队列中可以包含10个请求，每一个请求的语句序列如图7所示，而数据库的连接数为2，调度策略使用最简单的 FIFO。

等待队列中的10个最初的请求如图7所示。s1是等待队列中第一个位置的请求的第一个事务中第一个语句，根据 FIFO 策略，s1首先占有连接资源1，赋予它的 transactionID 值为1，然后把 transactionID 返回给该请求中这个事务的剩余语句，假设在这里 s2与 s1属于同一个事务，则语句 s2带着该 transactionID 值申请资源时，不用生成新的 transactionID；同理，s5是等待队列中第二个位置的请求的第一个事务中第一个语句，s5占有第二个连接资源，赋值 transactionID 为2，如图7i 所示；

第二个连接资源放回到连接池后，由于第一个连接资源上 s1语句尚未执行完毕，所以同一个事务体的语句 s2不参与竞争资源，s6是等待队列中第三个位置的请求的第一个事务中第一个语句，s6占有刚刚放回的第二个连接资源，赋值 transactionID 为3，如图7 ii 所示；

s9是等待队列中第四个位置的请求的第一个事务中第一个语句。当第一个连接资源放回到连接池后，为了避免请求饿死，s9占有第一个连接资源，赋值 transactionID 为4，如图7 iii 所示。

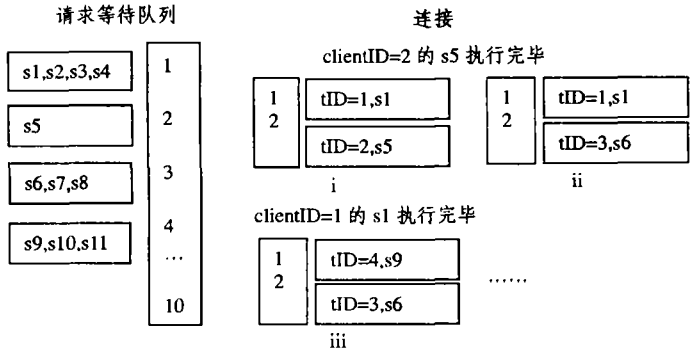

图7

6 在 PostgreSQL 上的实现

PostgreSQL 是一种对象-关系型数据库管理系统(ORDBMS),也是目前功能最强大,特性最丰富和最复杂的自由软件数据库系统。有些特性甚至连商业数据库都不具备。PostgreSQL 使用"一用户一进程"的 client/server 模型,一个 PostgreSQL 会话由下面一些 UNIX 进程(程序)组成:

·一个监控的守护进程 (postmaster);

·用户的前端应用(如 psql 程序);

·一个或者多个后端数据库服务器 (postgres 进程本身)。

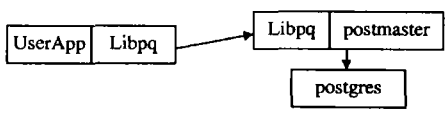

图8

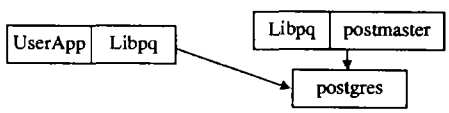

图9

一个 postmaster 管理某台主机上的一定的数据库集合。如果某个前端应用想访问某一数据库,它就会进行与应用链接到一起的接口库(函数)调用(比如 libpq)。该库把用户的请求通过网络发给 postmaster(图8),postmaster 接着便启动一个新的后端服务进程 postgres。并将前端进程和这个新的服务进程连接起来。从这时起,前端进程和后端服务将不再通过 postmaster 而是直接进行通讯(图9)。

PostgreSQL 使用"一用户一进程"的 client/server 模型。数据库服务器在初始化时可以指定连接数,即同时产生的 postgres 进程的最大值,显而易见,它也是同时处理的客户端应用的最大数。postgres 进程的个数最终将受制于操作系统的进程数,所以 postgres 进程数有限,这导致数据库服务器可以同时处理的用户请求数有限,因此,PostgreSQL 每分钟处理的新订单数 tpmC 的值比较小。

PostgreSQL 数据库的每一个连接的开始都是一个新的事务,一个连接包含一个或者多个事务;每个事务都有一个全局唯一的 transactionID,在恢复时,根据 transactionID 进行恢复。这种进程结构,符合本文提出的两种中间件模型的使用条件,可以应用我们的数据库连接中间件。目前我们正在 PostgreSQL 数据库上实现上述两种数据库连接中间件模型。

总结 数据库服务器对于客户端的请求都有一个最大的连接数,这限制了数据库每分钟处理的新事务数,即限制了数据库的 tpmC 的性能值。为了提高 tpmC 值,我们首先使用成熟的连接池的技术,降低建立和释放连接的开销,提高用户的响应时间。但是,连接池的技术并没有从本质上改变连接的管理和维护方法,连接池的调度单位是每个客户端的请求。要提升 tpmC 的性能,必须改变这种调度的策略,降低调度粒度。本文提出了通过分析每个客户端请求,以事务或者语句这样更小的粒度为调度单位的中间件模型,实现提高客户端的响应时间的目的。

本文对数据库服务器提出了两种中间件模型。其中,以事务为调度单位的中间件模型适用于所有的数据库服务器,而以语句为调度单位的中间件模型并不适用于所有的数据库服务器。对每个事务赋予唯一的 transactionID,并根据此 transactionID 进行恢复,有此特性的数据库服务器才可以使用以语句为单位的中间件。数据库 PostgreSQL 具有该特性,目前,这两种中间件模型在 PostgreSQL 上已经完成了设计。

在目前的模型中,优先级的调度策略使用了最简单的 FIFO。调度策略直接影响到中间件的性能,如何合理的调度已经执行了某些语句的事务、新的事务和新的请求,如何提高用户的响应时间,防止请求饿死,是下一步工作的重点。

以语句为调度单位的中间件技术,在本文的实现中,强制了同一个事务的语句只有在上一个语句执行完毕后,下一个语句才可以申请连接资源,这种模型在不改变数据库服务器的前提下,最大程度地提高了 tpmC 值;我们可以改变通信协议,可以使同一事务的语句带有事务内的顺序号,同时申请连接资源,数据库服务器需要检查语句的执行顺序,重新组合事务,才能执行,当然,这需要数据库服务器的支持。

这种中间件技术,可以在中间件层次实现事务的切换,在此模型的基础上,我们可以加大监督事务执行的力度,记录每个事务使用的资源,在中间件进行一次死锁预检测,减少事务执行中等待资源的时间,提高事务成功执行的概率,提高每个连接的使用效率。

参 考 文 献

1 仲萃豪. 中间件--构筑复杂分布式应用的关键技术. http://www.csdn.net/news/newstopic/2/2840.shtml
2 http://www.cn-java.com/target/news.php?news_id=1902
3 宣以广. 浅谈多层分布式体系技术与应用. http://www.huihoo.com/middleware/trade.html
4 李冕,张佐,吴秋峰. 数据库中间件的结构分析. 计算机应用技术,2001(1):23~26
5 蒙德龙,张美菊. 简述中间件 TUXEDO 技术. 计算机时代,2001(11):6~8
6 周园春,李淼,张建,李小欧,张飞. 中间件技术综述. 计算机工程与应用,2002(15):80~82