Postgresql 的 TPM 实现和改进

蒋亚琴[1],王红罡[1],李 沁[2]
(1. 同济大学,上海 201804;2. 上海证券交易所,上海 200120)

摘 要:传统的 DBMS 的性能随着连接数的增加会降低,而且商业的 DBMS 由于昂贵的价格而不适用于一般用户。虽然 Postgresql 性能逊于商业 DBMS,但是其性价比更高。文中旨在提高 Postgresql 的性能,在传统的服务器和客户之间增加 TPM,并增加对服务器的监测功能以及实现动态改变服务器和客户间的连接数。由于最佳连接数取决于服务器的负载状况,因此文中设计不同连接数的 TPM 来分析服务器状态。实验使用 TPC-C 标准,结果证明,该方法能有效提高 Postgresql 的性能,规定时间内的吞吐量有所增加,响应时间有所减小。

关键词:数据库;事物处理检测器;Postgresql
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2013)01-0021-04
doi:10.3969/j.issn.1673-629X.2013.01.006

Implementation and Improvement of TPM for Postgresql

JIANG Ya-qin[1], WANG Hong-gang[1], LI Qin[2]
(1. Tongji University, Shanghai 201804, China;
2. Shanghai Stock Exchange, Shanghai 200120, China)

Abstract: The performance of traditional DBMS will decrease when connection numbers between server and client increase. And the commercial DBMS is not widely used because of its expensive price. Though Postgresql has a part inferior to commercial DBMS on performance side, it has higher cost efficient. The objective is to realize high performance and reliability in DBMS. And propose a new way by adding TPM between server and client. The key points are the function to detect the server state and change the connection numbers dynamically. But the best connection number depends on the load state of server, so design TPM in various points to analyze server state. Use TPC-C benchmark in this test, and the results prove the effectiveness, and the throughout is bigger and responding time is shorter.

Key words: database; TPM; Postgresql

0 引 言

当今社会,数据库管理系统(DBMS)广泛地应用于电脑科技中。其中,Postgresql[1] 由于其许多功能的免费开源性而被许多客户垂青。但是现代数据的复杂性和海量性使得我们需要更高性能的 DBMS。Postgresql 某些性能逊于商业 DBMS,因此市场的占有率较低[2]。但是商业 DBMS 非常昂贵,而且不适用普通用户,因此试图改进 DBMS,使得其在低成本的情况下能够实现高负载。

文中通过实现 Postgresql 的改进来实现 DBMS 的高性能和高可靠性。Postgresql 存在随着连接数的增加,其性能会降低的问题,所以引入了事务处理检测器[3](Transaction Processing Monitor, TPM)来控制连接数,同时,对引入了 TPM 的系统进行评估。但是 TPM 存在连接如何管理的问题,文中针对此提出了有效的解决方案。

1 2 层数据库存在的问题

通常使用的 2 层数据库系统结构如图 1 所示。在

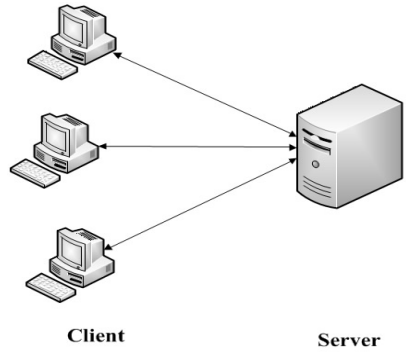

图 1 2 层 DBMS 结构

收稿日期:2012-05-10;**修回日期**:2012-08-14
基金项目:国家自然科学基金资助项目(61103069,71171148);国家科技计划课题(2012BAD35B01);上海市科技创新计划(11DZ1501700);上海信息化发展专项基金(20091015)
作者简介:蒋亚琴(1987-),女,硕士研究生,研究方向为数据库、数据挖掘。

该系统结构中,客户和服务器直接通信[4],一旦发生连接,Postgresql 便会生成一个进程。当连接过多时,服务器便会面临过载问题,从而使得性能降低。

因此,文中引入三层的 TPM 结构(如图 2 所示)。TPM 位于客户和服务器之间,解决了 Postgresql 中的过载连接问题,实现了服务器连接环境的最佳性。

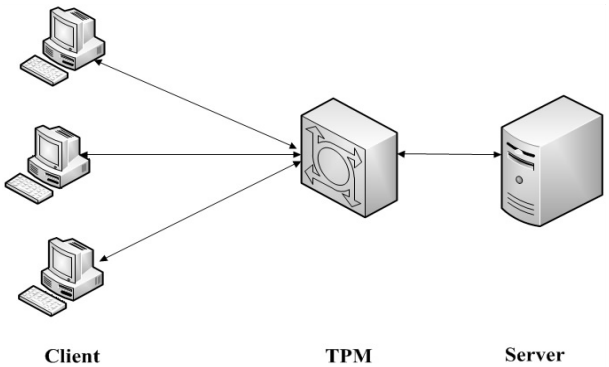

图 2 3 层 DBMS 系统

2 现有的 TPM

2.1 现有 TPM 的架构

TPM 的架构如图 3 所示。TPM 由线程和队列构成[5]。线程管理通信以及减少客户和服务器间的连接数。根据处理内容可将线程分为:

(1)客户线程:接受客户的连接和处理请求;

(2)服务器线程:和服务器通信;

(3)控制线程:调整连接数。

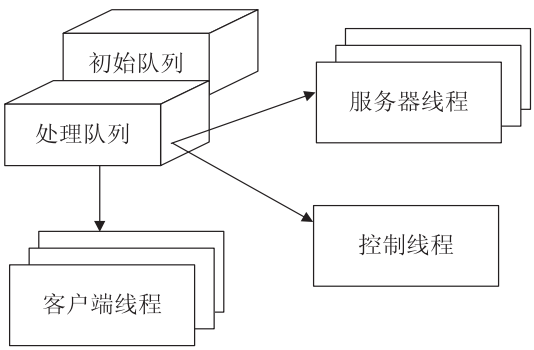

图 3 TPM 架构

队列用来存储请求,分为:

(1)初始队列:存储客户向服务器发出的连接请求;

(2)处理队列:存储客户的事务处理请求。

当与服务器的连接数超过设定值时,控制线程会接管客户的连接请求。因此,当连接数并未超过设定值时,控制线程不起作用。线程之间是并行运行的。一个进程会在服务器和客户之间拥有多个连接和通信。

由于客户和服务器间的连接数是不同的,所以使用队列来存储连接。连接请求有两种,两类队列分别存储对应的请求,同时,所有的线程共享队列数据。

2.1.1 连接问题的解决方法

首先介绍一下客户和服务器之间的连接处理过程。客户需要能够发送连接请求和接受应答信息[6]。所以,TPM 一开始将所有的客户转至已连接状态。连接处理流如下:

(1)客户线程接受客户的连接请求并将其放入初始队列中去。

(2)服务器线程从初始队列中取出一个请求并将其发送给数据库服务器。

(3)数据库服务器通过服务线程发送应答给客户,并转至连接状态。

当设定的服务器线程完成与服务器的通信,服务线程不再能够进行初始通信,连接请求保留在初始队列中[7]。因此,使用控制线程来处理该问题。控制线程的连接处理流:

(1)等待所有的服务线程结束初始通信。

(2)接受保留的连接请求。

(3)接受请求的控制线程,再启用服务器并将客户转为连接状态。

通过这种方法,所有的客户能够完成连接,同时服务器可以处理事务。

2.1.2 数据库处理流

当客户完成连接,便会发送处理请求,此时,数据库进入处理事务状态。数据库执行的是一系列的 select,add,delete 操作[8]。处理过程称作事务。数据库处理流如下:

(1)客户线程将初始请求插入处理队列中。

(2)服务线程将初始请求取出并发送给数据库服务器。

(3)服务线程和发送请求的客户通信,同时开始处理事务。

(4)当事务处理完成,系统又回到客户线程,转向(1)。

2.2 现有 TPM 存在的问题

现有 TPM 存在一个问题,即其每类线程的线程数人为设定,但是却不能动态改变。通过不同的标准评估过 TPM,发现服务线程的最佳数目是不同的,取决于服务器的大小和承载量。所以很有必要观察服务器的加载状况,而且 TPM 需要能够动态改变服务线程数。

3 TPM 的改进

3.1 服务器状态的监测功能

为了能够动态改变服务线程的数目,在 TPM 中增加了监测服务器状态的功能。通过在固定的时间间隔

内测量吞吐量以及通过在服务器运行期间不断降级[9]来监测 TPM。改进后的 TPM 如图 4 所示。

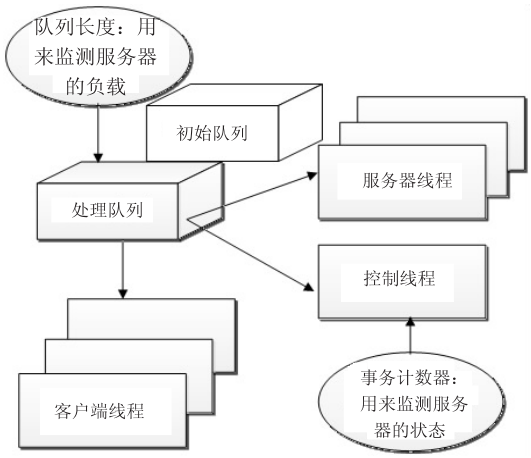

图 4　改进后的 TPM

3.1.1　事务计数功能

在固定的时间间隔内用吞吐量来监测服务器的性能,而且可以知道服务器运行时 TPM 的降级情况。

3.1.2　队列长度监测功能

通常情况下,当服务器加载量较低时,请求不会被放到队列中。因此,可以通过队列中存储的请求数(即队列长度[10])来监测服务器的处理性能。

3.2　连接数的动态改变

对于上述 TPM,初始连接数是固定的,在运行期间是不能动态改变的,但是,服务器的状态会影响最佳连接数。所以设计和实现了动态改变功能。设计内容主要分为两部分:操作线程和数据库的连接/断开。下面分别对其进行介绍。

3.2.1　操作线程

用标签[11]来管理服务线程的增加和删除,并设计了操作线程来管理该标签。操作线程的三个基本功能是:

(1)监管服务器的状态;

(2)管理标签;

(3)生成服务线程。

操作线程初始时会监控吞吐量和队列长度,并定期检查数据,然后通过监测到的信息来管理标签。若生成标签了,操作线程便会生成服务线程。操作线程的处理流如下:

(1)在固定时间里监测服务器;

(2)获得信息;

(3)若有新的标签生成,则操作线程会生成服务线程;若有标签被删除,或者目前状况已是最佳连接数,则转向 1。

3.2.2　数据库的连接和断开

服务线程是由操纵线程生成的,换言之,服务线程的删除[12]是在操作线程生成删除标签后自主完成的,与此同时,与服务器的连接也会断开,但这时,数据库的安全性就会出现问题。传统的断开方法是通过客户的断开请求来断开线程的通信。但是,运行时,当服务器的连接数发生改变,有可能会接受不到断开请求,所以,为了能在规定时间里结束线程,线程里加入了与服务器的断开操作,这样使得线程能够在任意时间自主断开与服务器的连接,同时,在运行时可以减少服务器的连接数。

当连接到服务器的数目增加时,需要生成新的线程,并且和服务器之间建立连接。以往的 TPM,当服务线程生成后,它会一直处于等待状态,直到已经连接到服务器的请求被插入初始队列。因此,数据库由于在等待运行中的连接请求而不能对事务进行处理。如今,改变了运行期间生成的连接,对于和数据库的连接,重新使用从客户那里收集到的连接请求。

每生成一个线程,已经设计好的计数器就会减少 1。当它变为 0 时,便会通知呼叫线程生成工作已经完成。而且,删除标签会在每次事务处理完成时被确认。

4　实验结果

尽管已经完成了服务器的连接数实现,但是根据状态来管理生成和删除标签的技术还未成功。在以往的测试中,最佳连接数会随着标准不同而变化,这就需要更加深入的观察。很明显,当集中式加载时,服务器的最佳连接数会减少。在这次测试中,为了监测集中加载的情况,定期观察队列长度,并将其与设定的阈值进行比较,从而通过比较结果动态地去改变连接数。本测试的环境如表 1。

表 1　测试环境

操作系统	RedHat Server 5
DBMS	Postgresql8.3.7
服务器	CPU/Intel(R) Xeon(R) Processor E5606 2.13GHz, Memory/ 8.00GB
TPM	CPU/Intel(R) Core(TM) i3 1.7GHz Memory/ 2.00GB
客户	CPU/Intel(R) Core(TM) i5 2.67GHz Memory/ 2.00GB

4.1　测试标准

使用 TPC-C[3]标准来衡量性能。当然,为了能够动态变化,对 TPC-C 也做了一些改变。在测试期间,TPC-C 的加载量是恒定的,而最佳连接数是不定的,为此在运行的时候会动态地改变加载量。

4.2　动态管理服务器连接数的方法

本实验中,使用队列长度作为动态改变服务器连接数的唯一依据。并由队列长度来设定阈值,同时通过当前队列长度和阈值的比较不断完成连接数的变化。操作线程的处理过程如下:

(1)每三秒观察一下队列长度。

(2)将队列长度和阈值进行比较,若队列长度是 40 及以上,那么 counter+1,若队列长度是 10 及以下,则 counter 值-1。

(3)如果 counter 值是 15,建立生成标签,增加一个服务器连接,并重置 counter 和标签值。

(4)如果 counter 值是-15,建立删除标签,删除一个服务器连接,并重置 counter 和标签值。

(5)返回 1。

在该实验中,如果队列长度是 40 及以上,便认为服务器已经过载,需要减少连接数。同理,若队列长度是 10 及以下,认为服务器还有很大的负载量,可以增加一些连接。最大连接数设为 10,最小连接数设为 5。

4.3 实验结果

使用 TPC-C 标准,2 小时内 TPM 吞吐量和平均应答时间如表 2 所示。在本实验中使用了 40 个数据仓库,400 个客户连接来测试,同时将服务器连接数分为固定和动态变化两种,固定的设置为 20,10 和 5。初始阶段服务器连接数设置为 20,并且在运行时比较固定连接数和动态改变连接数之间性能的不同。从结果来看,吞吐量提高了将近 4%,应答时间减少了大约 10%。

表 2 实验结果

连接数	20(固定)	10(固定)	5(固定)	动态变化
吞吐量(bps)	530.4	530.6	529.2	553.1
平均响应时间(秒)	5.98	6.03	6.01	5.39

5 结束语

文中通过改进 TPM 来提升 Postgresql 的性能,检测标准是 TPC-C,解决了现存 TPM 中三层数据库系统的问题,增加了连接数的动态变化和监测服务器状态的功能。

参考文献:

[1] 许宏松. PostgreSQL7 数据库开发指南[M]. 北京:机械工业出版社,2001:1-38.

[2] 林河水,程 伟,孙玉芳. PostgreSQL 存储管理机制研究[J]. 计算机科学,2004(12):76-80.

[3] 古 锐,元 伟,叶晓俊. 表空间存储策略 PostgreSQL 中的研究与实现[J]. 计算机工程,2006(16):38-40.

[4] Stonebraker M. The Design of the Postgres Storage System [C]//Proceedings of 13th International Conference on Very Large Data Bases. Brighton,England:[s. n.],1987.

[5] 陈文星,付继宗. Linux 下数据库 PostgreSQL 分析与应用[J]. 电脑开发与应用,2006(11):56-57.

[6] 吴 亮. 基于 PostgreSQL 的海量数据存储管理[D]. 长沙:中南大学,2005.

[7] 王 博,李 波,高振铁. 基于 TPM 的嵌入式可信计算平台设计[J]. 单片机与嵌入式系统应用,2011,11(1):13-16.

[8] 刘长浩,孙玉芳. PostgreSQL 请求优化机制研究[J]. 计算机科学,2005,32(4):163-167.

[9] 郭龙江,李金宝. PostgreSql 分析器研究[J]. 黑龙江大学自然科学学报,2001(4):50-52.

[10] Pacbev S. Understanding MySQL Internals[M]. [s. l.]: O´Reilly Media,Inc.,2007.

[11] 胡巧巧,王建名,叶晓俊. PostgreSQL 数据库预取算法研究[J]. 计算机科学,2006(6):138-139.

[12] 朱 江,沈庆国. 开放源码数据库 PostgreSQL 的特点及其应用实例[J]. 军事通信技术,2003(2):59-62.

(上接第 20 页)

参考文献:

[1] 陈智宇,温彦君,陈 琪. VxWorks 程序开发实践[M]. 北京:人民邮电出版社,2004.

[2] 孔祥营,张保山,俞烈彬. VxWorks 驱动及分布式编程[M]. 北京:中国电力出版社,2007.

[3] 周启平,张 杨,吴 琼. VxWorks 开发指南与 Tornado 实用手册[M]. 北京:中国电力出版社,2004.

[4] 周启平,张 杨. VxWorks 下设备驱动程序及 BSP 开发指南[M]. 北京:中国电力出版社,2004.

[5] 曹桂平. VxWorks 设备驱动开发详解[M]. 北京:电子工业出版社,2011.

[6] 张 杨,于银涛. VxWorks 内核、设备驱动与 BSP 开发详解[M]. 北京:人民邮电出版社,2011.

[7] VxWorks Reference Manual[M]. USA: Wind River System Inc,1997.

[8] VxWorks 5.5 Migration Guide 6.6[M]. USA: Wind River System Inc,2007.

[9] 周启平,张 杨. VxWorks 程序员速查手册[M]. 北京:机械工业出版社,2005.

[10] 李方敏. VxWorks 高级程序设计[M]. 北京:清华大学出版社,2004.

[11] 罗国庆. VxWorks 与嵌入式软件开发[M]. 北京:机械工业出版社,2003.

[12] Wind River. VxWorks Programmer´s Guide[M]. USA: Wind River System Inc,1997.

Postgresql 的 TPM 实现和改进

作者： 蒋亚琴， 王红罡， 李沁
作者单位： 蒋亚琴,王红罡(同济大学,上海 201804)， 李沁(上海证券交易所,上海 200120)
刊名： 计算机技术与发展
英文刊名： Computer Technology and Development
年，卷(期)： 2013(1)