第32卷 第20期 计 算 机 工 程 2006年10月
Vol.32 № 20 Computer Engineering October 2006
·软件技术与数据库· 文章编号：1000—3428(2006)20—0081—03 文献标识码：A 中图分类号：TP311

TPC-C 测试系统的实现

孙雪祥，蒋艳凰，张 怡
（国防科技大学计算机学院软件所，长沙 410073）

摘 要：TPC-C 是一种旨在衡量联机事务处理（OLTP）系统性能与可伸缩性的行业标准基准测试项目，它被全球主流软硬件厂商，数据库厂商公认为性能评价标准。麒麟(Kylin)是国产服务器操作系统。为了评估 Kylin 的性能，避免测试软件造成的性能瓶颈，需要一个严格符合 TPC-C 标准的测试软件，获取统一的性能指标，为 Kylin 的性能调优提供参考。目前开源的事务处理性能测试软件，大多用于测试系统某一方面的性能，很少有测试 OLTP 系统性能并严格符合 TPC-C 标准的开源测试软件，因此无法利用这些工具得到严格符合 TPC-C 规范的性能参数。而且由于测试结果的单一性和测试过程的不透明性，无法利用测试结果指导操作系统调优。该文在 Kylin 环境里，通过整合 Web 服务器软件 Apache、应用服务器 Tuxedo 和数据库管理系统 Oracle，设计并实现了一个 TPC-C 测试系统。该系统严格遵循 TPC-C 规范，减少了由于测试软件的不足而对测试结果的影响，为 Kylin 的性能调优提供了一定的参考。

关键词：TPC-C；Tuxedo；Apache

Implementation of TPC-C Benchmark Test System

SUN Xuexiang, JIANG Yanhuang, ZHANG Yi
(Software Institution, School of Computer, National University of Defense Technology, Changsha 410073)

【Abstract】 TPC Benchmark™ C (TPC-C) is an OLTP workload, which is considered as performance evaluating standard. Kylin is a server OS in Chinese with owned authority based on the POSIX standard. To evaluate Kylin's performance, a TPC-C testing system is needed. At present, most open source-test softwares, which attempt to implement OLTP are load test tools, and few of them accords with TPC-C benchmark. Hence, can't gain the performance metric for TPC-C. The singularity of the test results and the opacity of the test course make it difficult to improve the operatiing system. This article implements TPC-C benchmark with Tuxedo, Apache and Oracle on Kylin OS.This system strictly keeps to TPC-C benchmark, reduces the impact because of the shortage of the testing software, and gives suggestion for Kylin.

【Key words】 TPC-C; Tuxedo; Apache

TPC-C 测试规范中模拟了一个 OLTP 应用环境：一个大型商品批发商，它拥有若干个分布在不同区域的商品库和销售点，每个仓库负责为 10 个销售点供货，每个销售点为 3 000 个客户提供服务，每个客户平均一个订单有 10 项产品，所有订单中约 1%的产品在其直接所属的仓库中没有存货，需要由其他区域的仓库来供货。

该系统需要处理以下 5 种事务：New-Order(客户输入一笔新的订货交易)；Payment(更新客户账户余额)；Delivery(发货，模拟批处理交易)；Order-Status(查询客户最近交易的状态)；Stock-Level(查询仓库库存状况)。

TPC-C 的主要性能指标有：tpmC：每分钟处理的事务数目；TPC-C：3 年的总费用除以 tpmC，即 price/tpmC，也被称为性价比。

1 TPCC 测试系统结构设计

TPCC 规范规定，对服务器的性能测试通过对数据库处理访问请求的监视来实现。我们模拟一定数量的虚拟用户从浏览器访问服务端的数据库，服务器端处理请求并返回结果，客户端记录响应种类，时间及数量，从而测试服务器性能。

测试系统采用 3 层软件体系结构。服务器端运行操作系统 Kylin， 安装有 oracle 数据库，实现数据层；服务器端使用 Web 服务器 Apache 和中间件 Tuxedo 实现业务逻辑层；使用远程终端模拟器模拟大量访问仓库的用户，实现表示层。

测试系统结构如图 1 所示。

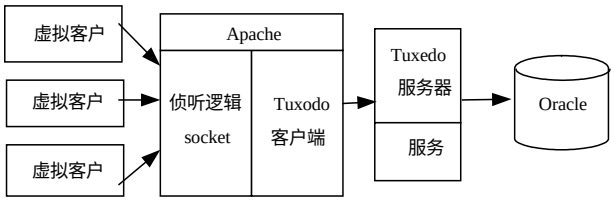

图 1 测试系统结构

1.1 测试数据库的设计

在服务器端的 kylin 操作系统上安装 Oracle，建立数据库 TPCC 作为待测数据库。按 TPCC 规范的要求，创建的数据库由 9 张基本表构成：Warehouse(仓库表)，District(街区表)，History(历史记录表)，Stock(库存表)， New-order(新订单表)，Customer(顾客表)，Item(商品表)， Order-Line(订单分录表)，Order(订单表)。数据库的大小由 Warehouse 的个数决定，每个 Warehouse 大约对应 100MB 数据。

为了提高数据库性能，建立测试专用的用户，并把表存储于各自独立的表空间。为了提高装载数据的速度，采用在服务器端通过 OCI 接口向数据库中装载数据的方式。

基金项目：国家“863”计划基金资助项目“服务器操作系统内核”(2002AA1Z2101)和“OS 新技术研究”(2003AA1Z2060)
作者简介：孙雪祥(1981－)，女，硕士生，主研方向：系统软件；蒋艳凰、张 贻，博士、副研究员
收稿日期：2005-12-28 **E-mail：**kujiangzhu@163.com

1.2 Tuxedo 中间件

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中间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，管理计算机资源和网络通信。Bea Tuxedo 属于交易中间件。交易中间件除具有跨平台、跨网络的能力外，它的主要功能是管理分布于不同计算机上的数据的一致性，协调数据库处理分布式事务，保障整个系统的性能和可靠性。交易中间件所遵循的主要标准是 X/Open DTP 模型。它适用于联机交易处理系统。

系统利用 Tuxedo 中间件把应用分为业务逻辑层、表示层和数据层 3 个不同的处理层。Tuxedo 中间件作为构造 3 层结构应用系统的基础平台，负责客户机与服务器间的联接和通信贷资金，实现应用与数据库的高效连接。

1.3 Apache

Apache 是因特网上使用最广的 Web 服务器软件。从实质上看，Web 服务器就是一个侦听网络上指定端口是否有客户连接的应用软件。Apache 最突出的优点表现在对附加模块的支持：Apache 的内部服务器引擎通过应用程序编程接口(API)向第 3 方程序开放。用户可以对 Apache 进行调整以满足自己的特定要求；并且，Apache 使得人们可以进行模块开发，免费获取并添加到服务器中，以扩充功能。

系统通过 Apache 实现捕获用户请求的功能，在 Apache 上加载了扩展模块 mod_tpcc，实现以下功能：实现事务标准输入输出界面；捕获客户端发送的请求，处理后交给 Tuxedo。

2 关键技术实现

2.1 Web 应用服务器 Apache

在 Apache 上加载的 mod_tpcc 模块，集成了捕获客户端请求和 Tuxedo 客户端的功能。虚拟用户向 Apache 发送一个 http 请求，mod_tpcc 模块捕获到请求，从 http 命令中抽取事务类型、仓库号、街区号等信息，根据这些信息调用 Tuxedo 客户端的请求服务函数，这些请求发送给 Tuxedo 服务器，Tuxedo 服务器响应服务后把结果返回给 Tuxedo 客户端，然后客户端返回 Apache，Apache 相应模块记录测试数据，显示处理结果。

本文在 Apache 上装载最稳定的多道处理模块(Multi-Processing Modules, MPM) prefork。perfork 用单独的子进程来处理不同的请求，进程之间是彼此独立的。prefork 的工作原理是这样的：控制进程在最初建立 StartServers 个子进程后，为了满足 MinSpareServers 设置的需要，创建一个进程，等待 1s，继续创建第 2 个，等待 1s，继而创建 4 个，如此按指数级增加创建的进程数，最多达到 32 个/s，直到满足 MinSpareServers 设置的值为止。这种模式可以使得不必在请求到来时再产生新的进程，从而减小了系统开销以提高性能。

MaxClients 设定的是 Apache 可以同时处理的请求数，这是对 Apache 性能影响最大的参数，其缺省值为 150，如果请求总数已达到这个值(可通过 ps－ef|grep httpd|wc－l 来确认)，那么后面的请求就要排队，直到某个请求已处理完毕。这就是为什么系统资源还剩下很多，而 http 访问却很慢的主要原因。150 的缺省值无法满足 TPCC 的测试需要，可以根据硬件配置和负载情况来动态调整这个值。Apache 默认的限制是不能大于 256。在 Apache2.0 中通过修改新加入的 ServerLimit 指令，使得无须重编译 Apache 就可以加大 MaxClients。

2.2 事务管理

事务，又称之为交易，指一个程序或程序段，在一个或多个资源如数据库或文件上为完成某些功能的执行过程的集合。分布式事务处理是指一个事务可能涉及多个数据库操作，分布式事务处理的关键是必须有一种方法可以知道事务在任何地方所做的所有动作，提交或回滚事务的决定必须产生统一的结果。

X/Open 组织(即现在的 Open Group)定义了分布式事务处理模型。X/Open DTP 模型(1994)包括应用程序(AP)、事务管理器(TM)、资源管理器(RM)、通信资源管理器(CRM)4 部分。一般，常见的事务管理器是交易中间件，常见的资源管理器是数据库，常见的通信资源管理器是消息中间件。

XA 是 X/Open DTP 定义的交易中间件与数据库之间的接口规范(即接口函数)，交易中间件用它来通知数据库事务的开始、结束以及提交、回滚等。XA 接口函数由数据库厂商提供。

TPCC 规范把顾客对系统的访问抽象成 5 种事务对数据库的访问。系统的资源管理器是 Oracle9i, 事务管理器使用交易中间件 Tuxedo，Oracle9i 与 Tuxedo 之间的连接遵循 XA 接口规范。因此，所有来自 Apache 的请求由 Tuxedo 负责管理。

本测试系统对 TPCC 规范规定的 5 种事务，采用事务操作定义和事务通信分开的方式。事务操作定义，即事务在数据库中要执行的所有 SQL 语句，都写入文件，存储在特定目录。客户端与数据库的连接是以事务进行连接的。Tuxedo 在 ubbtpcc 配置文件中配置好与连接数据库相关的选项，启动数据库及数据库侦听程序。启动 Tuxedo 服务器，中间件与数据库连接，发布服务：dy_transaction, pt_transaction, no_transaction, ol_transaction, sl_transaction。Apache 模块 mod_tpcc 通过 Tuxedo 的客户端程序调用服务器端发布的服务。所有的需要统计的数据，例如执行的事务种类，事务执行时间等，都记录在特定目录的日志文件和结果文件中。

2.3 模块间通信

2.3.1 Tuxedo 与 Oracle9i 之间的连接

Tuxedo 服务器在启动的时候通过读取配置文件连接数据库。Tuxedo 服务器启动后，发布 TPCC 规范规定的 5 种对数据库操作的事务，这 5 种事务通过 OCI 接口实现与数据库的连接。我们做了很多轮测试，通过不断地改变待测系统的设置和对测试结果的分析，发现在数据库方面，比较明显的影响测试结果因素包括：与数据库的连接方式和数据库中索引的建立。系统采用 OCI 接口访问数据库，提高中间件与数据库之间的通信效率，减少模块间通信对测试结果的影响。

应用程序访问 Oracle9i 有 3 种途径：OCI 接口，PRO*C，ODBC 接口。

(1)OCI(Oracle Call Interface)是由头文件和库函数等组成的一套 Oracle 数据库应用程序编程接口工具，OCI 程序实质上就是用高级语言写的程序，其特点是内部含有对 OCI 子函数的调用。OCI 程序对开发环境的要求相对较低，只要有 C 语言的 OCI 开发工具包和 C 编译器就可以，程序设计相对 PRO*C 复杂。

(2)PRO*C 是 Oracle 提供的应用程序专用开发工具，它以 C 语言为宿主语言，能在 C 程序中嵌入 SQL 语言，进行数据库操作。

(3)ODBC (Open DataBase Connectivity)即开放数据库互联，是一个用于访问数据库的统一界面标准，是应用程序和数据库系统之间的中间件。它通过使用相应应用平台上和所需数据库对应的驱动程序与应用程序的交互来实现对数据库的操作，避免了在应用程序中直接调用与数据库相关的操作，从而提供了数据库的独立性。

系统采用OCI接口来实现对数据库的访问。在以下方面OCI明显优于PRO*C：OCI提供动态绑定，可以绑定和定义包

括表在内的任意结构；OCI对任何服务器下的元数据的操作提供了详细的功能描述；OCI提供了增强的数组数据操作语言(DML)能力，允许数组的插入、更改和删除，在返回一批错误前完成尽可能多的迭代；OCI允许用户在一次执行时跟踪提交响应以减少往复(round-trips)。在以下方面OCI明显优于ODBC：OCI使查询最优化；透明的预取缓冲区减少了往复，提高了性能和可测量性，减少了服务器上的内存使用；OCI对往复做了优化；OCI是线程安全的，在任何OCI句柄中都不需要使用互斥，而ODBC不是线程安全的，需要手动保持很多数据结构的互斥；ODBC没有会话的概念；OCI减弱了连接，会话和事务的往复；多用户可以使用一个单独的连接，在对SQL操作的连接上进行序列化，每个用户可以多事务；ODBC的处理速度比OCI慢。

2.3.2 Tuxedo8.1 和 Apache 的连接

Tuxedo 和 Apache 的连接是通过函数实现的。在 Apache 上加载模块 mod_tpcc 时，直接把 Tuxedo 的客户端程序编译进该模块。当 Apache 接收到浏览器的请求信息后，mod_tpcc 模块直接调用 Tuxedo 的客户端程序，然后通过客户端程序调用 Tuxedo 服务器发布的服务。因为 mod_tpcc 模块集成了 Tuxedo 客户端的实现部分，所以在编译生成 mod_tpcc 模块时，必须在 makefile 里的编译语句中加上编译 Tuxedo 客户端的选项，而且要区分是编译成远程客户端还是本地客户端；在启动 Apache 之前，必须正确设置启动 Tuxedo 客户端的相关环境变量，否则，Tuxedo 客户端无法找到 Tuxedo 服务器。

3 实验结果

首先确定组件间连接和浏览器显示的输入输出页面符合 TPC-C 规范。然后启动客户端程序，对服务器进行访问测试。

本测试旨在证明 Apache +Tuxedo +Oracle 在 Kylin 上实现测试系统的正确性和本系统符合 TPC-C 规范，所以在两台普通 PC 上做测试。硬件测试环境：客户端和服务器端都是联想计算机，512MB 内存；通过以太网连接。服务器端软件：Kylin2.0 操作系统，Tuxedo8.1，Apache2.0。客户端软件：远程终端模拟器程序。服务器的数据库中灌了 10 个 WareHouse 的数据。测试结果如图 2。

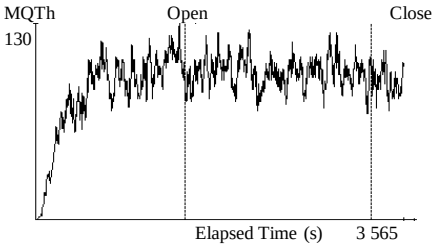

图 2 每分钟 New-Order 事务个数统计

图 2 中，横轴表示测试时间，竖轴表示每分钟事务数(tpmC)。Open 和 Close 之间是测试稳定时间。在此次测试中，Kylin 的 tpmC 值为 130。因为取测试点的时间间隔过长，所以性能曲线不光滑。

4 结论

本文主要阐述用 Apache +Tuxedo +Oracle 在 Kylin 国产服务器操作系统上实现 TPC-C 测试系统，在设计和开发过程中尽量避免测试工具对测试结果造成的影响，得到客观的测试结果。本文构建的 TPCC 测试系统为 Kylin 操作系统的性能调优提供了一定的参考。

参考文献

1 徐春金. Tuxedo 中间件可发与配置[M]. 北京: 中国电力出版社, 2003.

2 Aulds C. 马树奇, 金 燕译. Linux Apache Web Server 管理指南[M]. 北京: 电子工业出版社, 2001-03.

3 Garcia-Molina H, Ullman J D, Widom J. 杨冬青, 唐 世, 徐其钧译. 数据库系统实现[M]. 北京: 机械工业出版社, 2001-03.

4 刘少锋. Linux 数据库编程[M]. 北京: 人民邮电出版社, 2001-08.

5 fei@cnfug.org. FreeBSD 下 Apache2.0 运行模型分析及性能调整[Z]. 中文 FreeBSD 用户组, http://www.cnfug.org.

(上接第 80 页)

这里值得一提的是本文没有将接口合约的后置条件实现在测试驱动代码中，而是利用在每一个测试用例中放置的测试期望值来检查合约后置条件的满足与否。原因有两个：(1)接口中不同的方法对于不同的测试用例后置条件的判断条件是不同的，接口合约中描述的有些后置条件由于构件信息隐蔽的因素无法在测试驱动代码中实现；(2)有些后置条件的检查很难完成，比如返回客户自定义类的判断，必须使用特殊的检查类或者采用手工的方式才能完成检查任务。

图 4 接口合约产生向导

3.3 测试驱动产生器

目前设计的测试驱动产生采用简单的读写文件方式，即根据给定的测试用例，调用接口合约检查产生器产生测试代码，然后产生实际的方法调用代码。方法较简单，不再赘述。

4 结论

本文提出了一种基于合约检查的构件化软件集成测试框架，通过该框架可以高效地运行测试用例，检查出构件软件集成测试中的问题，并且能够准确地定位错误位置。通过对原型系统的开发，可以充分证明 CCTF 框架是一种切实可行的构件化软件集成测试策略。

参考文献

1 Meyer B. Object-oriented Software Construction [M]. Prentice Hall, 1997.

2 Adrita B. Software Component Testing Strategies[R]. Dept. of Information and Computer Science, University of California, 2001.

3 Yoonsik C, Leavens G T. A Runtime Assertion Checker for the Java Modeling Language(JML)[C]. Proc. of Software Engineering Research and Practice. CSREA Press, 2002: 322-328.

4 Jerry G. Component Testability and Component Testing Challenges[C]. Proceedings of Sofware Testing, Analysis and Review 1999.

5 Mariani L. Behavior Capture and Test: Dynamic Analysis of Component-based Systems[D]. Milano Bicocca: Universit`A Degli Studi Di, 2005.