

# J2EE 架构下连接池技术的应用与改进

侯 灿, 杨宗凯, 刘 威

(华中科技大学 电子与信息工程系, 湖北 武汉 430074)

**摘 要:**在现代企业信息管理和电子商务中,数据库技术被广泛应用于对企业数据进行有效管理,其中一个重要的研究课题是如何有效访问位于企业系统的后端数据库。介绍了 J2EE 架构下直接连接和池连接访问两种方式,并比较它们的性能优劣,然后在此基础上,针对远程客户机的访问对池连接方式作了进一步改进,提出通过本地缓存连接池引用,来改善远程客户机数据库交互的性能。最后使用仿真程序考察 3 种不同的数据库访问方式。测试结果证明带缓存的池连接访问确实能够为远程访问数据库带来明显的性能提升。

**关键词:**J2EE;JDBC;连接池

**中图分类号:**TP311.131

**文献标识码:**A

**文章编号:**1673-629X(2006)10-0008-03

## Application and Improvement of Database Connection Pool Based on J2EE Architecture

HOU Can, YANG Zong-kai, LIU Wei

(Department of Electronics and Information Engineering, Huazhong University of Science & Technology, Wuhan 430074, China)

**Abstract:** The database technology is widely used in enterprise information management, of which the effective access to database is one important point. This paper introduces two ways of database access in J2EE architecture: direct connection and pool connection, and makes a comparison of their performance. Then it analyses remote clients' access procedure and makes improvement by buffering reference of connection pool in local address. At last, the simulation experiment of three ways of database access is made, which proves that buffered pool reference does promote the remote client's performance of accessing database.

**Key words:** J2EE;JDBC;connection pool

### 0 引言

在采用 J2EE 技术的企业级电子商务系统中,通常采用三层体系架构的系统应用模型<sup>[1]</sup>,即客户机层、中间层和后端层。其中,中间层的主要作用是提供高效率的访问后端数据库服务器的方法。如果数据库访问方法选择不当,就会使得数据库访问效率低下,影响整个系统的运行效率。

目前常见的数据库访问方法是直接连接,然而,其系统开销大,易导致内存泄漏,从而维护困难。为此,人们提出了采用连接池技术来提高数据库连接效率。文中针对 J2EE 架构下数据库访问的特点,对连接池方式进行了深入的分析,并在此基础上进行了改进,提出带缓存的数据库访问连接池。

### 1 J2EE 架构下的数据库访问方法

在基于 J2EE 架构的应用系统中,对数据库的访问从本质上来看都是通过 JDBC(Java Database Connectivity, SUN 公司制定的基于 Java 的标准的、统一的数据库访问机制)实现的。但从客户机层访问渠道以及性能的角度考虑,则需根据 J2EE 客户机层的不同应用渠道,通过不同的上层程序如 JSP 或 Servlet 及其它客户应用程序,使用 JDBC、连接池与数据库进行交互,进而达到存储和管理数据的目的。

#### 1.1 J2EE 客户机层对数据库访问的要求

J2EE 架构的系统应用模型中,对数据库访问的要求开始于客户机层,客户机层的类型不同导致了数据库访问技术的不同。最常见的两种客户机类型为 WEB 客户机和独立客户机。

WEB 客户机即 WEB 浏览器,使用 HTTP 协议和应用服务器交互,通过 JSP,Servlet 等 Web 组件进行数据库访问。WEB 方式访问数据库的特点是突发性强,峰值访问数和平均访问数相差较大,在某一时刻对数据库的访问量会很大、并发请求的数量也会很多,而保持数据库连接

收稿日期:2006-02-13

基金项目:国家十五科技攻关项目(2004BA205A27-1)

作者简介:侯 灿(1982-),女,湖北黄石人,硕士研究生,研究方向为电子商务;杨宗凯,教授,博导,博士,研究方向为电子商务、网络教育、智能信号处理与应用、宽带网络通信技术。

的时间很短,交互数据量也很小。

独立客户机是用 Java 或其他语言编写的在客户本地机器上部署的程序,其通过标准的 socket 库、使用中间层提供的 Java 命名和目录接口(Java Naming Directory Interface, JNDI)的命名服务和 JDBC 来进行数据库访问。采用这种方式的企业商务系统通常使用客户端进行大规模的数据访问应用,如电子数据交换、报表导入导出等,其特点是需要在这段时间内保持稳定的数据库连接,每次数据库的访问量都很大。

应用系统所采用的数据库访问技术,必须能够满足以上两种主要客户渠道的要求。下面分别对 JDBC 直接连接、池连接等几种 J2EE 架构下使用的数据库访问方法进行原理和性能分析,然后对性能较优的池连接方法进一步改进,得到改进后的池连接方法。

### 1.2 基于 JDBC 的直接连接

基于 JDBC 的访问方法通过 JDBC 驱动程序建立与数据库的物理连接,访问操作结束后断开连接,具体操作有 5 个步骤<sup>[2]</sup>:

步骤 1:加载驱动程序与数据库建立连接;

步骤 2:创建一个 JDBC 声明;

步骤 3:执行 SQL 语句,并将结果存储到一个数据集中;

步骤 4:接收并处理数据集中的记录内容;

步骤 5:关闭创建的对象和连接。

直接连接方式存在很多问题。首先,每次请求操作都将建立一个数据库连接,并直接与数据库管理系统通信。当服务器的访问量很大、并发请求的数量很多时,这种方式将大大增加系统开销。其次,所建立的连接必须被正确关闭,对使用这些连接的程序提出了一定的要求。如果出现程序异常而导致某些连接未能关闭,将导致数据库系统的内存泄露和死机。为了解决直接连接的这些问题,出现了数据库连接池的技术。

### 1.3 普通池连接访问

池连接是一个维护数据库连接的内存缓冲区,由 J2EE 应用服务器进行管理和维护。其工作原理是<sup>[3]</sup>:应用服务器启动时,在连接池中建立一定数量的数据库连接,随时为用户的请求分配池中空闲的连接,连接使用完毕后,立即释放。与数据库系统的物理连接的建立与断开都由连接池管理。其工作原理如图 1 所示。

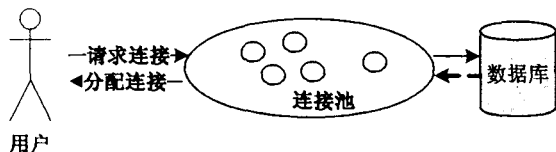


图 1 连接池基本工作原理图

在 J2EE 架构下,客户机需要使用 JNDI 服务来查找连接池并获得连接<sup>[4]</sup>。客户机采用池连接方式的具体步骤为(如图 2 所示):

步骤 1:和服务器建立一个 JNDI 的上下文环境;

步骤 2:通过 JNDI 名称查找连接池,把连接池的对象引用保存在本地;

步骤 3:向连接池请求数据库连接;

步骤 4:创建一个 JDBC 声明;

步骤 5:执行 SQL 语句,并将结果存储到一个数据集中;

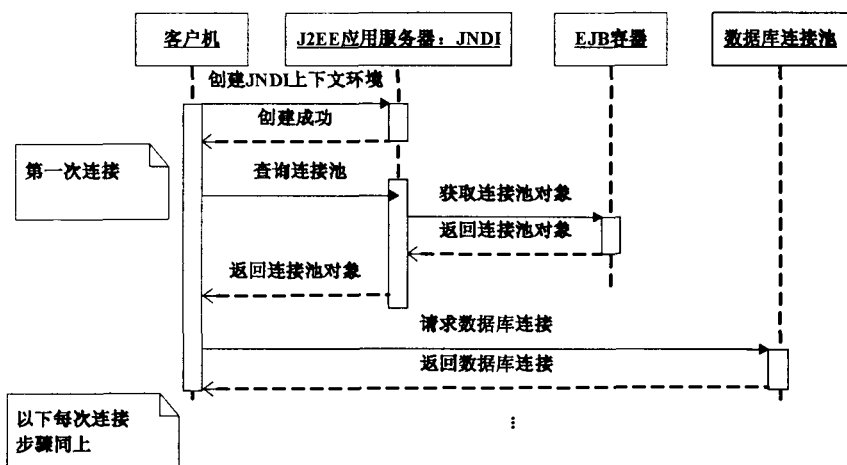


图 2 J2EE 架构下的池连接方式获取数据库连接时序图

步骤 6:接收并处理数据集中的记录内容;

步骤 7:关闭创建的对象和连接;

步骤 8:关闭连接池的对象引用和 JNDI 上下文环境。

数据库连接池将访问数据库的连接作为一种资源进行管理。当用户需要访问数据库时,连接池分配已有连接给客户程序,而不需要重新请求建立数据库连接;当用户访问完数据库之后,连接池释放掉相应连接资源即可,也不必关闭至数据库的物理连接。这样,到数据库的物理连接数被稳定在一个较少的数量上,数据库连接的使用效率大大提高,建立和释放连接的开销减少<sup>[5]</sup>。

普通池连接访问基本满足了 WEB 访问的要求。因为 WEB 应用中的访问组件由 J2EE 应用服务器管理,所需要的 JNDI 查询操作在 JVM 中直接调用执行,执行效率较高。但是,普通池连接方式不适用于远程客户机,因为 JNDI 查询是客户机和服务器在网络上的来回交互,其性能受到网络传输质量的影响。为此,这里提出了一种改进的池连接方式。

## 2 带缓存的池连接访问

分析普通池连接访问方式,发现其中存在 4 次交互:

交互 1:客户机程序和 J2EE 应用服务器建立 JNDI 查询的上下文环境;

交互 2:客户机程序向 J2EE 应用服务器查询连接池对象,应用服务器返回该对象的引用;

交互 3:应用服务器从 EJB 容器中获得连接池对象;

交互 4: 客户机程序向连接池请求数据库连接, 连接池返回可用的数据库连接。

交互 3 属于 J2EE 服务器 JVM 上的直接方法调用, 而交互 1、交互 2、交互 4 都是网络交互过程, 其中交互 4 对于任何池连接方式都是必需的, 而另外两个交互并不是必需的。客户机程序每次向 J2EE 服务器查询的是同一个连接池对象, 如果客户机程序在第一次查询连接池对象后, 把该对象引用缓存在本地, 以后的每次数据库操作就可以直接使用该对象来请求数据库连接。这样, 客户机程序在第一次获取连接池对象后, 从第二次开始池连接访问方式为:

- 步骤 1: 向连接池请求数据库连接;
- 步骤 2: 创建一个 JDBC 声明;
- 步骤 3: 执行 SQL 语句, 并将结果存储到一个数据集中;
- 步骤 4: 接收并处理数据集中的记录内容;
- 步骤 5: 关闭创建的对象和连接。

图 3 为在 J2EE 架构下以优化池连接方式获取数据库连接时序图。

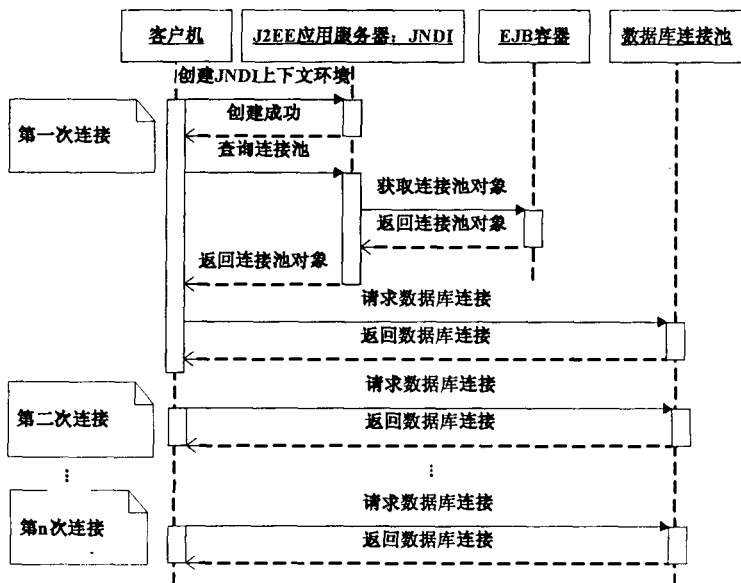


图 3 J2EE 架构下的优化池连接方式获取数据库连接时序图

这一方式省去了建立 JNDI 上下文的交互和 JNDI 查询连接池的交互, 因此也就省去了这两次 JNDI 操作的时间。优化后通过连接池完成数据库访问的部分代码如下:

```

//客户机程序启动时.....
//初始化 JNDI 上下文环境
context = InitialContext(J2EEServerURL);
//查询连接池对象
connectionPool = context.lookup(Pool JNDIName);
.....
//数据库访问
//获取连接
connection = connectionPool.getConnection();
//执行 sql 操作

```

```

connection.executeSql(sql);
//释放连接
connection.close();
.....
//客户机程序关闭时
context.close();
connectionPool = null;

```

### 3 各种访问方式性能比较

采用仿真程序考察以上 3 种不同的数据库访问方式 (JDBC 直接访问, 普通池访问方式, 带缓存的池访问方式) 的性能差异。测试环境为同一局域网里的 3 台主机, 一台 (P4 CPU 3.00GHz, 1G RAM) 部署了 Weblogic8.14 服务器, 一台 (P4 CPU 3.00GHz, 1G RAM) 部署了 Oracle9i 数据库, 另一台 (P4 CPU 2.00GHz, 512M RAM) 运行客户程序向服务器请求连接。测试程序设定了进行数据库访问 (一次访问规定为获取连接并即刻关闭) 的次数, 程序循环发起对数据库的访问请求, 并记录每种方式获取  $n$  ( $n = 1, 10, 100, 200, \dots, 1000$ ) 次连接的耗时, 测试结果如图 4 所示。

由此可见, 连接池的使用明显优于传统的数据库直接连接。而采用了缓存连接池又在非缓存连接池的基础上对性能进行了进一步的优化, 平均用时节省 16.6% ((非缓存用时 - 缓存用时) / 非缓存用时)。而且, 需要提出的是: 由于测试环境是局域网, 所以从服务器获取连接池的时间并不长, 如果客户端和服务端之间经由公众网络, 则带缓存的池连接方式可带来的性能提升将更为明显。

### 4 结 论

提出一种改进的数据库连接池访问方式, 通过本地缓存连接池引用, 有效地改善了以远程序方式访问数据库的性能。

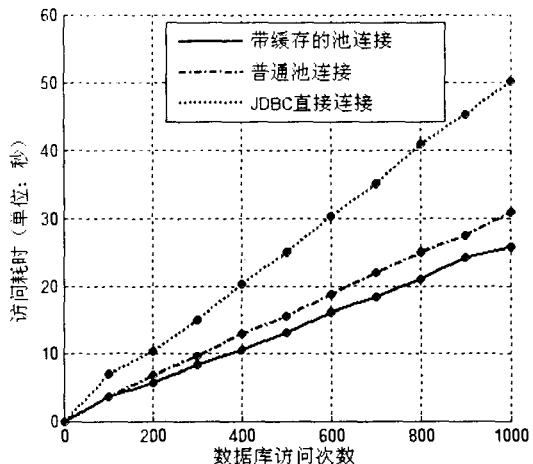


图 4 J2EE 架构下的数据库访问方法性能比较图

(下转第 13 页)

得到兴趣词条  $p_i$  的权重和新鲜度后,可以根据公式  $t_i = w_i \times f(x_i)$  计算词条  $p_i$  的兴趣度,式中  $f(x)$  为词条新鲜度对权重的影响函数。

词条兴趣度是网络服务搜索特性分析的最终依据。

### 1.3 异构服务搜索的特性分析

现举一个例子来说明异构服务搜索的特性分析处理过程。

假设已经得到了用户的兴趣字典,它由若干兴趣词条与相应词条的权重以及新鲜度组成,根据这些词条与相应词条权重、新鲜度构造用户个人兴趣树。兴趣树的根节点是总类,叶节点是兴趣字典中的词条,其他节点可能是兴趣字典中的词条也可能不是(这依赖于兴趣字典中词条是否存在蕴涵关系)。从根节点到叶节点概念上越分越细。每一个节点由词、权重、新鲜度三部分组成。兴趣生成树中每一中间节点词的权重按公式(3)计算,每一中间节点词的新鲜度按公式(4)计算<sup>[5]</sup>。

$$\text{Node}(p_j) \cdot w_j = \sum_{i=1}^k w_i \cdot w_i \quad (3)$$

$$\text{Node}(p_j) \cdot x_j = \sum_{i=1}^k \frac{w_i x_i}{w_i} \quad (4)$$

式中  $w_j$  为中间节点  $p_j$  的权重,  $x_j$  为中间节点  $p_j$  的词条新鲜度,  $k$  为节点  $p_j$  的子节点个数,  $w_i$  为子节点兴趣词条  $p_i$  的权重,  $x_i$  为子节点兴趣词条  $p_i$  的新鲜度。

在获得中间节点的权重与新鲜度后,可以根据公式  $t_i = w_i \times f(x_i)$  计算每一节点的兴趣度。如果一个节点的兴趣度越大,则认为用户对此节点中包含的内容兴趣越大。因此当用户输入检索词存在一词多义情况下,可以通过比较几种含义在用户个人兴趣树中兴趣度来选择用户真正感兴趣的含义。

以 interest 关键词的查询为例,假设已建立兴趣搜索树,如图 2 所示(图中所标注的兴趣度已经被单位化)。interest 一词有两种含义:一是经济术语“利润”;二是文化上的“兴趣”。那么用户是想搜索“利润”的含义还是“兴趣”的含义,可以通过分析用户个人兴趣树来选择。在用户个人兴趣树中,用户对经济的兴趣度为 0.15,对文化的兴趣度为 0.08。由于经济的兴趣度比文化的兴趣度大,因此认为用户对经济类型的东西更感兴趣,因而系统认为用户实际查询的是“利润”含义的。实际上,这种方法是按兴趣度预先将用户划归为不同兴趣类型,然后再根据用户兴趣类型选择相应检索词含义,以达到特性分析的目的<sup>[6]</sup>。

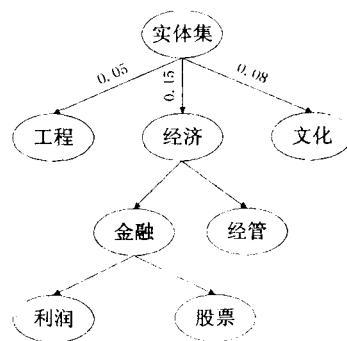


图 2 兴趣搜索树

## 2 结论及相关研究

综上所述,与其他需求分析工具相比,文中所叙的需求分析器具有以下优点:由于备有个性化的服务搜索引擎,可以协助用户更加恰当地表达需求,从而精确、迅速地从 Internet 上搜索到所需要的信息,并且有效地避免了无关信息的干扰;同时,在需求分析与服务搜索交互的基础上,能够为需求分解粒度的控制提供参考。

异构服务组平台需求分析工具的研究是一个涉及面相当广泛的课题,它包括应用需求的表示工具研究、应用需求的功能分解以及分解建模后的运行流程的表示工具研究等。后续研究包括:需求分析建模工具的集成、需求分解粒度的精确控制研究、需求分析的成果—运行流程的形式化推导以及运行流程的文档化表示等。

### 参考文献:

- [1] 李雪梅. 网络搜索的个性化服务[J]. 中国信息导报, 2003(3): 26-27.
- [2] 胡 坤. 等待第三代搜索引擎[J]. 电子商务世界, 2005(8): 40-44.
- [3] 戴建中. GnetFtp 搜索引擎的算法设计与实现[J]. 汕头大学学报(自然科学版), 2005(3): 69-74.
- [4] Ozan E. Virtual reality in requirement analysis for CIM system development suitable for SMEs[J]. International Journal of Production Research, 2002(7): 3693-3708.
- [5] Merunka V. Object-oriented approach in requirement engineering for the analysis of information systems[J]. Journal of Forest Science, 2005(3): 13-18.
- [6] Drake J M. Approach and case study of requirement analysis based on acquisition ontology[J]. International Journal of Intelligent Systems, 2000(4): 1125-1155.

(上接第 10 页)

### 参考文献:

- [1] Artiges M. BEA Weblogic Server 8.1 Unleashed[M]. 北京: 机械工业出版社, 2005.
- [2] 王爱冬, 阳国贵, 张 涛. J2EE 架构下的数据库访问技术分析与研究[J]. 齐齐哈尔大学学报, 2005, 21(3): 39-42.

- [3] 杨 瑞, 蔡 虹. 连接池技术及其 java 实现[M]. 应用技术, 2003(6): 26-29.
- [4] 黄 文, 谢寄石. 基于 J2EE 的数据库连接服务[J]. 电子科技大学学报, 2002(2): 68-71.
- [5] 陈梅容, 郭 俊, 朱兵章. 在 JSP 中采用连接池技术优化数据库连接[J]. 机电工程技术, 2004, 33(4): 59-60.