基于线程池的高性能服务器软件的设计和实现

胡 萍[1], 陈志鹏[2]
(1.浙江大学 宁波理工学院,浙江 宁波 315100;
2.中国联通杭州分公司,浙江 杭州 310000)

摘 要:传统的多线程技术,虽然在提高服务器性能发挥了重要作用,但是在实时系统应用中,由于其频繁的创建和销毁线程的系统开销严重影响了系统性能,因此有效降低这部分开销对进一步提高服务器性能尤为重要。文中在分析传统服务器模型的弊端基础上,提出了基于线程池的服务器模型,并分析说明了线程池技术如何提高服务器性能,最后给出了应用C++语言实现该服务器模型的具体方法。

关键词:线程池;C++;高性能;服务器
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2006)08-0049-02

Design and Realization of Software of High-Performance Server Based on Thread Pool Technology

HU Ping[1], CHEN Zhi-peng[2]
(1.Ningbo Inst.of Technology,Zhejiang Univ.,Ningbo 315100,China;
2.China Unicom Branch Company in Hangzhou,Hangzhou 310000,China)

Abstract:Though traditional multithread technology is improving server performance and playing an important role, but in the real-time system, because the system expenses of its frequent establishing and destroying thread have influenced systematic function seriously, so it is particularly important to reduce this part of expenses effectively to further improve server performance. This paper has proposed the server model based on thread pool on the basis of analyzing the drawback of the traditional server model, analyzed and proved how thread pool technology raise server performance, provided the method of implementing this server model using C++ finally.

Key words:thread pool; C++; high-performance; server

0 引 言

在很多实时系统中对响应时间都有较高的要求,如数据库系统、订票系统等等。传统多线程服务器软件采用的模型是一旦接收客户端任务请求时,立即创建一个新的线程来执行任务,任务执行完毕后,线程退出[1]。尽管创建线程比创建进程,系统的开销已经小多了,但是如果提交给线程的任务执行时间较短的话,那么服务器将忙于创建和销毁线程,而服务器处理客户端请求任务的时间就相对减少了,这样服务器的性能就受到很大的影响。文中提供一种解决方案来解决以上问题。

1 基于线程池的高性能服务器软件的设计思路

服务器线程的执行分为3个过程:线程创建时间(T1),线程执行时间(T2),线程销毁时间(T3)。如果线程执行时间(T2)很短的话,那么服务器创建和销毁线程的开销(T1+T2)是不可忽视的[2],当它占用服务器开销超过一定比率的时候,将会严重影响服务器的性能。因此,要提高服务器资源的利用率,应着眼于减少T1和T2这两部分额外的开销。采用线程池技术能有效地解决这个问题。

线程池提供了处理系统性能和大用户量请求之间矛盾的方法,通过对多个任务重用已经存在的线程,降低了对线程创建和销毁的开销,由于当客户请求到达的时候,线程对象已经存在,服务器就能立即处理客户请求,从而提高了服务器的响应能力。线程池采用预创建技术,即在服务器软件启动时,创建一定数量的线程,放入空闲队列中,这些线程处于阻塞状态(线程处于阻塞状态不占用CPU,只占用很少一部分内存)[3]。当客户端任务到达时,从线程池中选择一个空闲的线程执行任务。与传统的多线程服务器模式不同,线程池中的线程执行完任务后并不退出,而是处于阻塞状态,等待下一个任务到达。线程池中线程的数目是动态调整的,当线程池中的所有线程都处于繁忙状态时,线程池自动创建新线程来处理更多的任务;当线程池比较空闲时,线程池就销毁一部分空闲线程,

收稿日期:2005-11-23
作者简介:胡 萍(1978-),女,湖北人,硕士,助教,研究方向为数据库技术及应用。

释放系统资源。

线程池至少包含线程池管理器、工作线程、任务接口等部分。其中线程池管理器负责创建、销毁和调度工作线程;工作线程是循环执行任务的线程,在没有任务的时候进行等待;任务接口是每个任务必须实现的接口,主要用来规定任务的入口、任务执行完后的清理工作和任务执行状态等等,而工作线程是通过调用这个接口来执行具体任务的。

线程池框架主要包括 4 个类:TaskBase(任务接口),Thread(线程基类),WorkThread(工作线程类),ThreadPool(线程池管理类)。线程池中各个类之间的关系如图 1 所示。

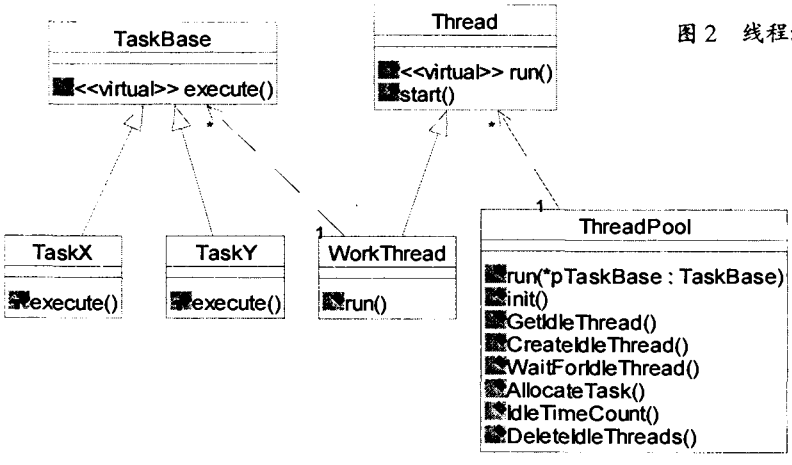

图 1 线程池中各个类之间的关系图

其中任务接口类 TaskBase 是个抽象的接口,其派生类必须实现 execute()方法,工作线程通过执行这个 execute()方法来执行具体任务;线程基类 Thread 是对操作系统提供线程接口的封装,它是所有工作线程类的基类,提供统一的接口 run()方法,所有派生类都要实现 run()方法,同时 Thread 还提供了 start()方法,它封装了启动线程细节;工作线程类 WorkThread 从 Thread 类派生,实现了 run()方法,工作线程启动以后将执行这个 run()方法;线程池管理类 ThreadPool 负责管理线程池,包括创建、销毁、调度工作线程等等;TaskX 和 TaskY 都从 TaskBase 派生,在 execute()方法中定义了具体的任务内容。

2 基于线程池的高性能服务器软件的具体实现

线程池的引入,不仅有效解决了服务器线程开销的问题,提高了服务器性能,还带来了一个简洁、直观、易于扩展的多线程服务器模型,提高了代码的可读性和可维护性[4]。线程池在服务器应用程序中的应用模型图如图 2 所示。

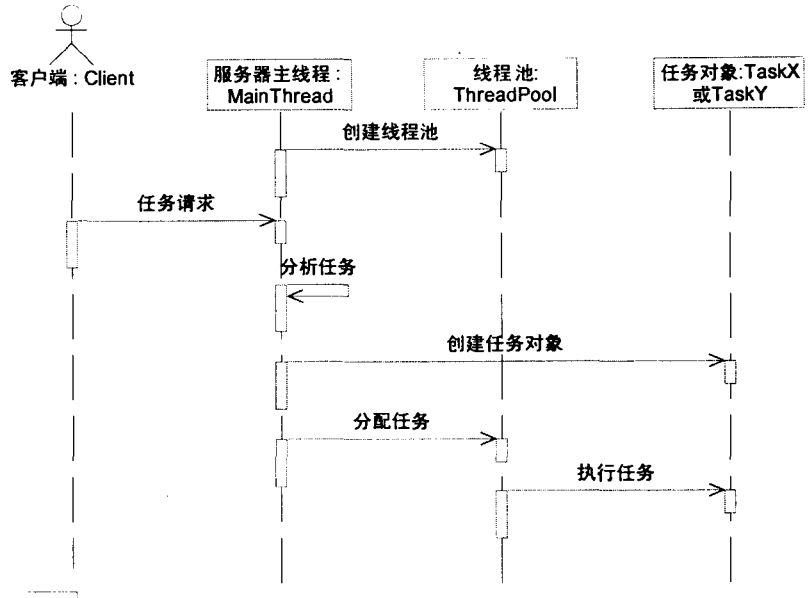

图 2 线程池在服务器应用程序中应用的交互图

其中服务器主线程首先创建并初始化线程池,并等待客户端请求;当有任务请求到达时,服务器主线程接收并分析任务,根据任务类型创建相应的任务对象;然后服务器主线程将任务交由线程池执行,而服务器主线程继续等待下一个客户端请求[5]。

其服务器软件代码框架如下所示:

```
void main()
{
  //创建并初始化线程池,THREAD_NUM 为初始化线程池的工作线程数
  ThreadPool * pThreadPool = new ThreadPool (THREAD_NUM);
  while (1)
  {
//循环从客户端接受任务,解析任务,判断任务类型
switch(AcceptTask(TaskData))
{
case 任务类型 1:
  //执行任务 1
  //创建相应的任务对象 TaskX
  TaskX * pTaskX = new TaskX();
  //设置任务数据
  …
  //分配任务,由工作线程执行任务
  pThreadPool - > run(pTaskX);
  break;
case 任务类型 2:
  //执行任务 2
  //创建相应的任务对象 TaskY
  TaskY * pTaskY = new TaskY();
  //设置任务数据
  …
  //分配任务,由工作线程执行任务
  pThreadPool - > run(pTaskY);
```

(下转第 54 页)

是正定的当且仅当 K^* 是条件正定的。

将(12)式带入(11)式:存在 c.p.d 核 K^* 使得:

$$|| \phi(x) - \phi(x^T) ||^2 = \frac{1}{2}(K^*(x,x) + K^*(x^T,x^T) - K^*(x,x^T) \quad (13)$$

特别的,如果 $K^*(x,x) = 0$(如在 Hilbert 空间表示下),(13)式可简化为:

$$|| \phi(x) - \phi(x^T) ||^2 = - K^*(x,x^T) \quad (14)$$

同时另一方面,有

$$(\phi(x) - \phi(x_0), \phi(x^T) - \phi(x_0)) = \frac{1}{2}[- || \phi(x) - \phi(x^T) ||^2 + || \phi(x) - \phi(x_0) ||^2 + || \phi(x_0) - \phi(x^T) ||^2] \quad (15)$$

因而可以通过使用 c.p.d 核来代替直接计算核函数的分级内积,这样做的另一个优势是:由于内积运算没有变换不变性,而 $|| \cdot ||^2$ 运算具有平移等变换不变性,因而可以具有更大的适用范围。

事实上[5],任何负的平方距离核函数均是条件正定而非正定的,即如下形式的核[4]均是 c.p.d. 核(而非 p.d. 核):

$$K^*(x,x^T) = - || x - x^T ||^\beta \quad (16)$$

Smola 等人证明[5]只有 $0 < \beta < 2$ 的(16)式才适用于构造非线性分类算法。

4 实验及分析

为了验证文中所述算法的有效性,笔者使用标准测试集:鸢尾属植物数据集(iris.dat)进行了分类实验。在实验中分别使用多项式核函数和径向基核函数对数据集的4种属性(萼片长度、萼片宽度、花瓣长度、花瓣宽度)的3个类别(setosa, versicolor, virginica)进行 KPCA 和 KDDA 分析,引入 $\beta=1$ 的条件正定核后,利用该类型核的组合执行上述核分类算法,并对其分类性能进行对比测试,结果如图1所示。实验结果表明使用该 c.p.d 核与多项式核具有等价的效果。由于采用的 c.p.d 核是线性距离核,其整体的分类效果不如使用径向基核。

由于 KPCA 和 KDDA 在图像处理、语音和信号处理、模式识别、机器学习等领域有着广泛的用途,上述实验也表明使用 c.p.d 核同样能够完成这类非线性分类和识别的任务。

5 结束语

随着模式识别与机器学习技术的发展,及越来越多复杂分类任务的提出,使得简单线性分类算法或其组合均难以胜任这种要求。核策略等非线性方法的使用有效地扩展了学习算法。通过对条件正定核的使用,说明了基于核距离的非线性算法较传统距离度量有更强的适应性和有效性,比 Mercer 核有着更宽松的约束条件,因而有效地扩展了核方法的适用领域。

参考文献:

[1] 孙即祥.现代模式识别[M].长沙:国防科技大学出版社,2002.

[2] Duda R O,Hart P E,Stork D G.模式分类(第2版)[M].李宏东,姚天翔译.北京:机械工业出版社,2003.

[3] Lu Juwei. Face Recognition Using Kernel Direct Discriminant Analysis Algorithms[J]. IEEE TRANSACTIONS ON NEURAL NETWORKS,2003,14(1):117-126.

[4] 孔 锐,张国宣.基于核的K-均值聚类[J].计算机工程,2004(11):12-13.

[5] Smola A J,Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion [R]. GMD Technical Report,GMD- FIRST,Berlin:[s.n.],1998.1064-1097.

(上接第50页)

```
        break;
    }
  }
  //释放线程池内存
  delete pThreadPool;
}
```

3 结束语

线程池是组织服务器应用程序的有用工具,可以显著改善服务器程序的性能,在服务器领域有着广泛的应用前景。文中在分析了传统的多线程服务器软件处理轻量级任务的弊端的基础上,提出了一个基于线程池技术的服务器软件模型;同时给出了一个通用的可扩展线程池框架,以及其在服务器软件中应用的具体实现。基于线程池技术的服务器软件模型,对于提高需要处理轻量级任务并且实时性要求较高的服务器性能有很大帮助,具有现实意义。

参考文献:

[1] Beveridge J,Wiener R. Win32多线程程序设计[M].侯 捷译.武汉:华中科技大学出版社,2002.

[2] 王险峰,刘宝宏.Windows环境下的多线程编程原理与应用[M].北京:清华大学出版社,2002.

[3] Butenhof D R. POSIX多线程程序设计[M].于 磊,曾刚译.北京:中国电力出版社,2003.

[4] Hughes C,Hughes T. C++面向对象多线程编程[M].周良忠译.北京:人民邮电出版社,2001.

[5] Richter J. Windows高级编程指南[M].王书洪,刘光明译.北京:清华大学出版社,1999.