

# 基于线程池的数据采集器在网管中的应用

宋开卫, 石 坚, 程 哲

(华中科技大学 电子与信息工程系, 湖北 武汉 430074)

**摘 要:**针对网络管理中对采集器的实时性和可扩展性的要求,设计并实现了一个基于线程池的高效网络管理数据采集器。其中详细讨论了它的各项关键技术——Java 5 线程池、类的动态装载和消息队列,以及由此所带来的高效性、通用多用途性、模块化、可扩展和可移植等特性。该采集器应用于综合网管中,正确采集到 Oracle 数据库服务器 9.0.1 的 SNMP 接口的管理数据和某 PON(无源光网络)设备上的管理信息,较之以前,在性能上有了很大的提高。

**关键词:**线程池;动态装载;采集器;网络管理

**中图分类号:**TP393.07

**文献标识码:**A

**文章编号:**1673-629X(2007)07-0210-03

## Application of Threadpooled Collector in Network Managerment

SONG Kai-wei, SHI Jian, CHENG Zhe

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

**Abstract:** An efficient collector based on the threadpool is designed and implemented, which is applied to network management which requires the real-time and extensible performance. Its key technologies such as thread-pool in Java 5, dynamic loading a class, message queue, which make the collector efficient, all-purpose, modularized, extensible and replantable, are discussed in detail. It has correctly collected SNMP data of Oracle database server 9.0.1 and also management information of some passive optical network equipment, with performance greatly improved comparing with the past.

**Key words:** threadpool; dynamic load; collector; network management

## 0 引 言

随着 Internet 的迅猛发展,为了能够提供可靠而灵活的网络服务,网络管理与监控越来越成为一个突出的问题。被管理网络变得越来越庞大,网络提供的服务也越来越多样化,因而对网管服务器采集和处理数据的实时能力的要求也越来越高。

网络数据采集一般采用多线程技术来提高采集效率。但是在网络管理中,需要频繁地处理用户请求而每次请求需要的服务时间又很简短。此时,系统会将大量的时间花费在线程的创建与销毁上。Java 5 的线程池克服了这些缺点。通过对重用线程来执行多个任务,避免了频繁的线程创建与销毁的开销,能使得采集服务器的性能得到很大提高<sup>[1,2]</sup>。文中就是利用 Java 5 中的并发线程池技术,设计了一个通用的多用途数据采集器,并应用在综合网管中,正确采集到 oracle 数

据库服务器 9.0.1 的 SNMP 接口的管理数据和某 PON(无源光网络)设备上的管理信息,较之以前,在性能上有了很大的提高。

## 1 Java 5 并发集合工具包 java.util.concurrent 的简介

### 1.1 三个新加的多线程工具包

Java 5 中新加入了三个多线程包:java.util.concurrent, java.util.concurrent.atomic, java.util.concurrent.locks。

(1)java.util.concurrent 包含了常用的多线程工具,是新的多线程工具的主体。

(2)java.util.concurrent.atomic 包含了不用加锁情况下就能改变值的原子变量,比如说 AtomicInteger 提供了 addAndGet()方法。Add 和 Get 是两个不同的操作,为了保证别的线程不干扰,以往的做法是先锁定共享的变量,然后在锁定的范围内进行两步操作。但用 AtomicInteger.addAndGet()就不用担心锁定的事了,其内部实现保证了这两步操作是在原子量级发生的,

收稿日期:2006-09-08

作者简介:宋开卫(1975-),男,山东人,硕士研究生,研究方向为网络管理、系统优化;石 坚,教授,研究方向为网络管理、网络系统优化。

不会被别的线程干扰。

(3)java.util.concurrent.locks 包含锁定的工具。

## 1.2 新的任务执行架构

在 Java 5.0 之前启动一个任务是通过调用 Thread 类的 start()方法来实现的,任务的提交和执行是同时进行的,如果你想对任务的执行进行调度或是控制同时执行的线程数量就需要额外编写代码来完成。5.0 里提供了一个新的任务执行架构使你可以轻松地调度和控制任务的执行,并且可以建立一个类似数据库连接池的线程池来执行任务。这个架构主要由三个接口和其相应的具体类组成。这三个接口是 Executor, ExecutorService 和 ScheduledExecutorService,其关系如图 1 所示。

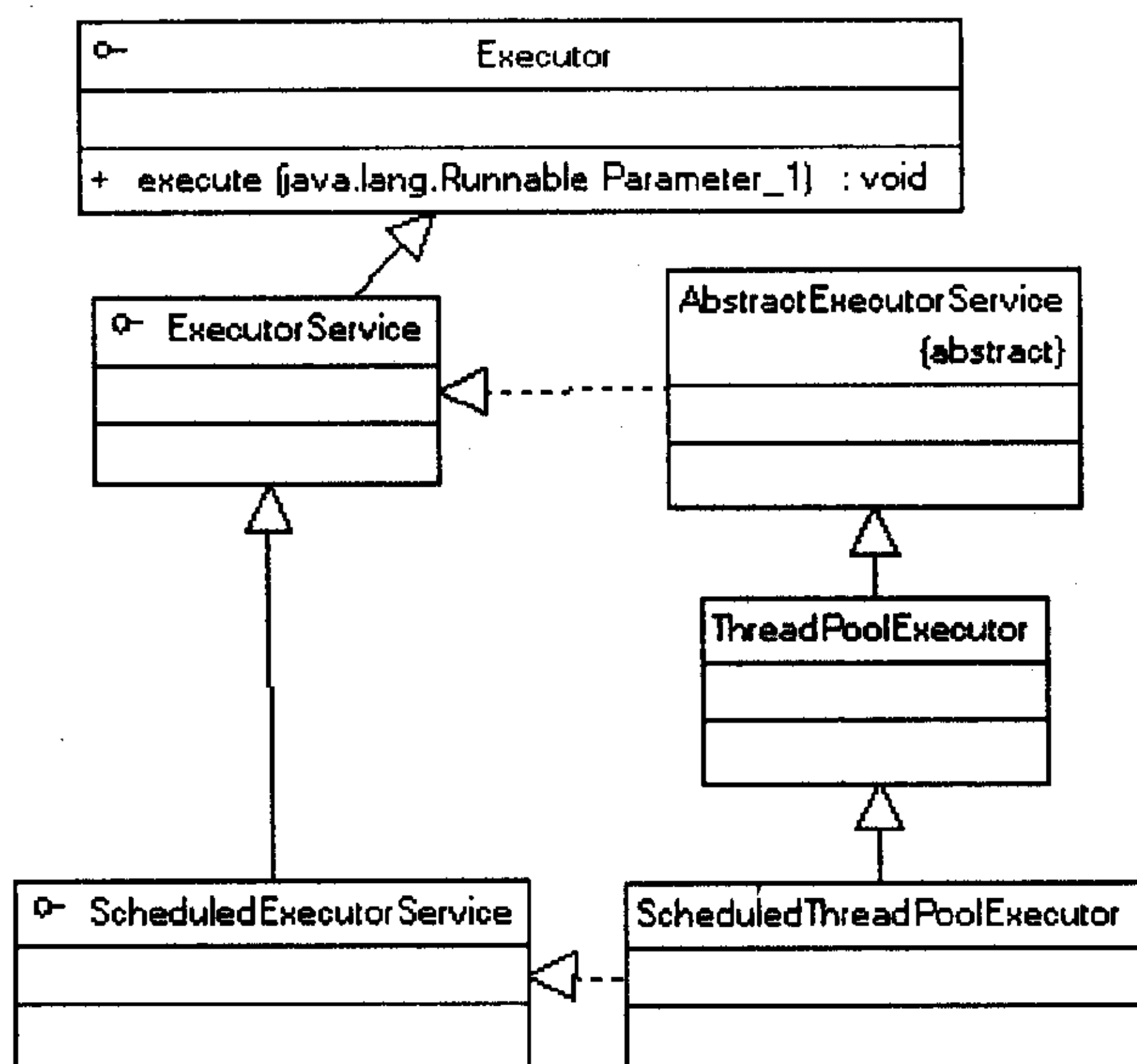


图 1 线程池接口关系图

图的左侧是接口,图的右侧是这些接口的具体类。注意 Executor 是没有直接具体实现的。

(1)Executor 接口:是用来执行 Runnable 任务的,它只定义一个方法——execute(Runnable command),其功能是执行 Runnable 类型的任务。

(2)ExecutorService 接口:它继承了 Executor 的方法,并提供了执行 Callable 任务和中止任务执行的服务。

(3)ScheduledExecutorService 接口:它在 ExecutorService 的基础上,提供了按时间安排执行任务的功能。

(4)Executors 类:

虽然以上提到的接口有其实现的具体类,但为了方便,Java 5.0 建议使用 Executors 的工具类来得到 Executor 接口的具体对象,需要注意的是 Executors 是一个类,不是 Executor 的复数形式。Executors 提供了以下一些 static 的方法:

①newCachedThreadPool():产生一个 ExecutorSer-

vice 对象,这个对象带有一个线程池,线程池的大小会根据需要调整,线程执行完任务后返回线程池,供执行下一次任务使用;

②newFixedThreadPool(int poolSize):产生一个 ExecutorService 对象,这个对象带有一个大小为 poolSize 的线程池,若任务数量大于 poolSize,任务会被放在一个 queue 里顺序执行;

③newScheduledThreadPool(int poolSize):产生一个 ScheduledExecutorService 对象,这个对象的线程池大小为 poolSize,若任务数量大于 poolSize,任务会在一个 queue 里等待执行。

在我们设计的网管采集器中,就是利用 Executors 类的 newScheduledThreadPool(int poolSize)方法产生一个具有足够线程数的线程池来执行各种各样的数据采集任务。较之一般的多线程技术,线程池主要用于单个任务处理的时间很短而请求的数目却十分巨大的场合,以消除频繁的创建和销毁线程的开销,从而达到提高实时性能的目的。除了创建和销毁线程的开销之外,活动的线程也消耗系统资源。在一个 JVM 里创建太多的线程可能会导致系统由于过度消耗内存而用完内存或“切换过度”。为了防止资源不足,服务器应用程序需要一些办法来限制任何给定时刻处理的请求数目。

线程池为线程生命周期开销问题和资源不足问题提供了解决方案。通过对多个任务重用线程,线程创建的开销被分摊到了多个任务上。其好处是,因为在请求到达时线程已经存在,所以无意中消除了线程创建所带来的延迟。这样,就可以立即为请求服务,使应用程序响应更快。而且,通过适当地调整线程池中的线程数目,也就是当请求的数目超过某个阈值时,就强制其它任何新到的请求一直等待,直到获得一个线程来处理为止,从而可以防止资源不足。

由于线程池是与业务无关的,并且经常会遇到诸如同步错误和死锁等并发风险,以及资源不足和线程泄漏等风险,自己从头开发定制的线程池并没有优势可言,因此利用 Java 并发工具包这样成熟的线程池技术是应用开发的首选。

## 2 采集器的设计

采集器是综合网管服务器软件中的一个模块,主要负责采集网络中被管设备或下层网管中的管理数据,管理数据也通过采集器配置到被管设备或下层网管中。考虑到采集器模块的相对独立性以及将来可能独立部署成一个综合的采集服务器,设计了网管服务器与采集器之间明确的通信接口——JMS 消息中间

件,其中设计了 Server 消息队列(网管服务器向采集器发送消息 Bean 格式的采集或取消命令)和 Collector 队列(采集器把采集到的数据转换成与采集协议无关的统一格式的消息 Bean 并发送给网管服务器)。通过 JMS 消息中间件,实现了采集器和网管服务器的解耦。有关 JMS 中间件的介绍可参见文献[3]基于 JMS 的数据推送系统的设计与实现。

整个采集器模块框图如图 2 所示。包括一个消息收发器线程、一个消息处理器以及 RunnableJob 接口。

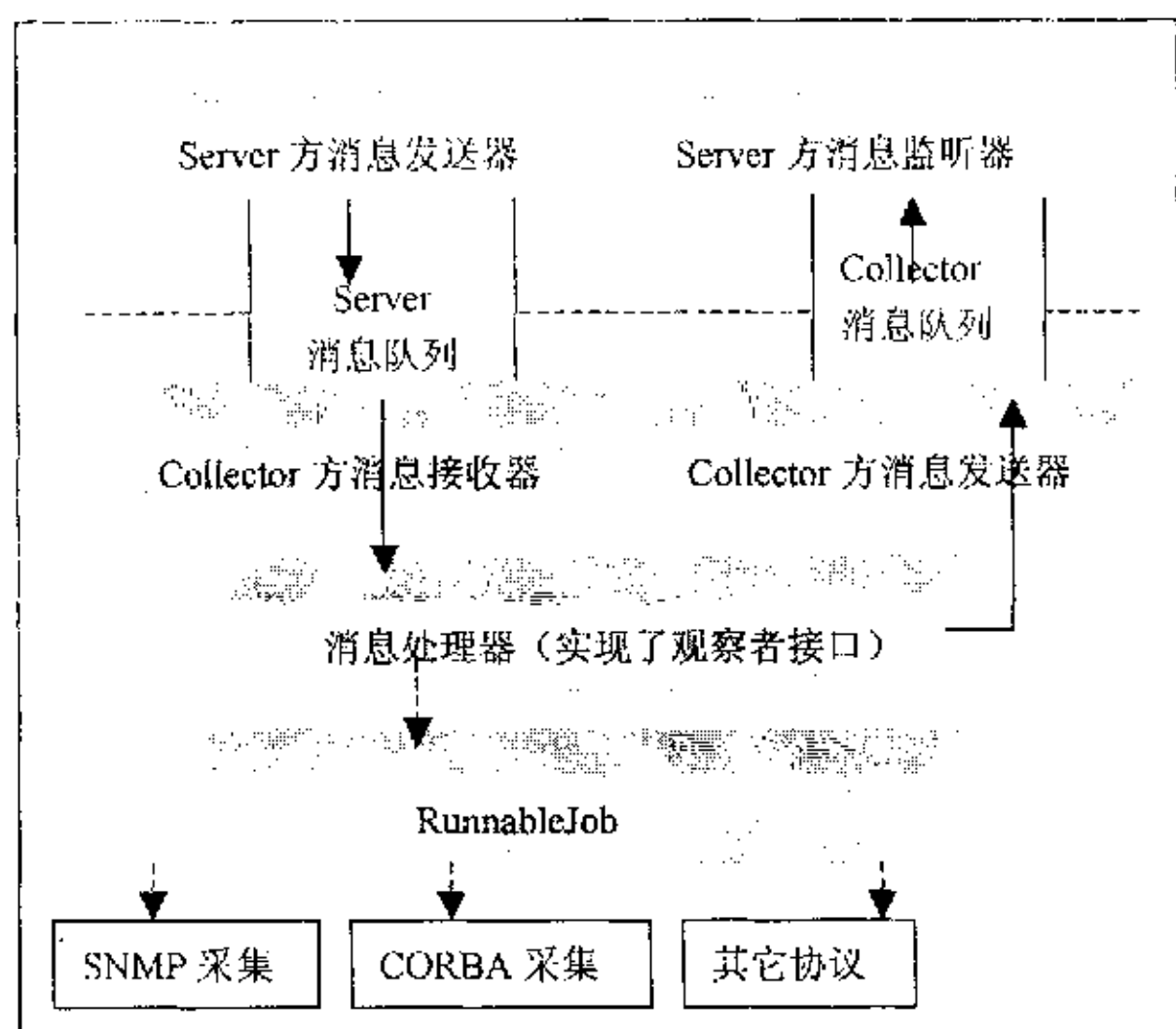


图 2 采集器设计框图

消息收发器比较简单,主要负责消息的接收并通知消息处理器,以及消息的发送。

消息处理器实现了观察者接口,负责消息的处理、通过线程池调用 RunnableJob 启动采集,并调用消息收发器的发送方法向网管服务器返回采集到的数据。由于采集常常涉及到一些轮询的工作,需要频繁地处理用户请求而每次请求需要的服务时间又很简短,所以非常适用线程池来处理。

RunnableJob 接口继承了 Runnable 接口,对应于线程池中的一个线程,负责具体的采集工作,并转换成网管服务器所要求的与具体采集协议无关的消息。

需要注意的是,实现 RunnableJob 接口是网管服务器一方的任务,采集器一方只是预装了各种各样的采集包,而并不需要了解网管服务器一方在运行时到底采用哪一个采集包。这里用到了 Java 类的动态装载机制。

Java 类的动态装载机制是 JVM 的一项核心技术,可以在不影响系统其它功能模块正常运行的同时动态地加载或替换系统的某些功能模块<sup>[4]</sup>。装载通过寻找一个类或一个接口的二进制形式来构造代表这个类或这个接口的 class 对象,其中类或接口的名称已给定。

多数情况下,JVM 使用类路径机制来搜索你在代码中引用的但尚未载入运行时的类。类路径是系统用来寻找类文件的一系列位置列表。但 Java 类的动态装载机制的更强大之处是它能够从对应用程序有意义的地方例如一个远程计算机载入类<sup>[5]</sup>。我们的采集器中,消息处理器就是通过在运行时从网管服务器传过来的 RunnableJob 的类字节码来动态创建它的对象的。

### 3 采集器的特点

(1)高效性。基于线程池技术的采集器,消除了线程创建和销毁的开销,提高了采集效率。

(2)通用性。由于 RunnableJob 是与采集无关的,所以采集器的通用性表现在两个方面:第一,采集器可以适用于各种各样的采集任务,目前在我们的网管服务器的设计中,针对不同的被管网络或下层网管就实现了 SNMP、CORBA、Socket 等协议,实现了网络管理有关的采集任务。第二,在我们的网络管理中,“Collector”表现为采集器(池),而也正是由于 RunnableJob 是与采集无关的,所以,任何适用于线程池的场合,都可以应用“Collector”,“Collector”还可以表现为 Web 服务器的连接池、并行计算池等等。

(3)通过 JMS 消息队列实现与 Server 的隔离。

### 4 结束语

提出了一种基于线程池的通用数据采集器的实现方案,并应用于综合网络管理软件的设计开发中,取得了较好的效果。该方案主要包括一个消息收发器线程、一个消息处理器以及 RunnableJob 接口。该设计方案利用了 Java 类的动态装载技术、线程池技术以及消息中间件 JMS,实现了通用多用途的高效的数据采集,并实现了与网管服务器的松散耦合,具有相对的模块独立性,为以后的扩展和移植提供了便利。

#### 参考文献:

- [1] 王 华,马 亮,顾 明. 线程池技术研究与应用[J]. 计算机应用研究,2005(11): 141-142.
- [2] 胡 萍,陈志鹏. 基于线程池的高性能服务器软件的设计和实现[J]. 计算机技术与发展,2006,16(8): 49-50.
- [3] 汪红兵,余春东,范植华,等. 基于 JMS 的数据推送系统的设计与实现[J]. 计算机应用,2005(S1): 366-368.
- [4] 黄晓平. Java 虚拟机中类装载机制的原理与应用[J]. 软件导刊,2006(1):10-12.
- [5] Arnold, Gosling, Holmes. The Java Programming Language [M]. Third Edition. 虞万荣,王玉峰,赵 亮译. 北京:中国电力出版社,2003: 242-249.