代理模式在数据交换资源调度中的应用①

胡晴晴[1,2]，孙 咏[2]，唐洪刚[3]

[1](中国科学院 研究生院，北京 100049)
[2](中国科学院 沈阳计算技术研究所，沈阳 110171)
[3](沈阳市安监局 安全生产应急救援指挥中心，沈阳 110034)

摘 要：在数据交换系统的任务设计中，资源调度问题直接影响整个系统的执行效率。因此，编写一个执行效率高，可维护性和可扩展性强的调度系统对于交换系统是很重要的。很多的调度系统中资源在队列中阻塞，但是线程却有空闲的情况频繁出现。很多的时候调度资源种类的增加会给程序员带来很大的麻烦.这是因为软件的耦合度高，小的改动会波及很多的部分。在本文中将代理模式和工厂模式等应用到资源调度中去，降低了软件耦合度，建立了一个可维护性和可扩展性强，执行效率高的调度系统。

关键词：资源调度；代理模式；设计模式

Application of Proxy Patterns to Data Exchange Resource Scheduling

HU Qing-Qing[1,2], SUN Yong[2], TANG Hong-Gang[3]

[1](Graduate University, Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110171, China)
[3](Shenyang Safety Supervision Bureau, Safety Emergency Rescue Command Center, Shenyang 110034, China)

Abstract: In the design of data_exchange ETL, the problem of Resource Scheduling influence the whole system's efficiency directly. So it is very important to write an efficient, easy-to-maintenance, and scalability system. There are resources to block in the case of frequent idle threads in many other scheduling systems. Always deployment of resources will increase a great deal of trouble to programmers. Because of software coupling, small changes will affect many parts of system. The small change will affect a lot of parts. This article uses the proxy pattern, the factory pattern to be applied to resource scheduling which reduced the coupling of software. We established a strong maintainability and scalability, the implementation of efficient scheduling system.

Key words: resource scheduling; proxy pattern; design patterns

随着计算机处理能力爆炸式的发展，人们所研究的软件系统也越来越庞大和复杂，早在上个世纪的六七十年代就已经有了软件危机的说法。而今，人们对软件扩展性、可复用性和可维护性的要求也越来越强烈。在这样的需求驱使之下，软件界有了 OCP 的软件设计原则。所谓的 OCP(Open Closed Principal)原则即“开—闭”原则就是：软件应对扩展开放，对修改封闭。正是为了满足软件设计中的 OCP 原则，设计模式的说法就应运而生了[1]。

设计模式（Design pattern）是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性[2]。设计模式在最初提出的时候只有几种，如今已经发展到了几十种。选择合适的设计模式用于我们的软件开发中，会带来事半功倍的作用。在本文的项目开发中就选择了代理模式等设计模式。

本文的选题的项目背景是国家水体污染控制与治理科技重大专项“辽河流域水环境风险评估与预警平台建设及示范研究”课题的数据交换系统。数据交换系

① 基金项目:国家水体污染控制与治理科技重大专项(2009ZX07528-006-05)
收稿时间:2011-07-15;收到修改稿时间:2011-08-22

统中任务资源调度是整个系统能否正常高效运行的最重要的环节。

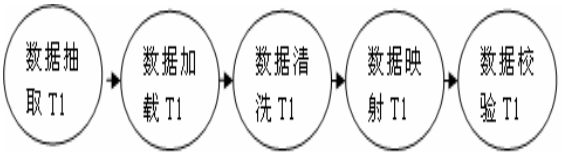
1　传统的数据交换资源调度模式

1.1 数据交换平台

数据交换平台是在多个数据终端设备（DTE）之间，为任意两个终端设备建立数据通信临时互连通路的过程。在辽河流域的水环境系统中，为了能及时安全的共享和交换各个监测点的信息和数据，建立一个数据交换平台势在必行。在数据交换系统中 ETL（即任务设计）的设计是不可缺少的部分。资源调度是任务设计最重要的部分。

1.2 资源调度概述

任务由一系列的作业组成，而作业又由一些顺序既定的资源组成。在一个任务中资源是最细粒度的执行单位。这里所说的资源是指对数据的基本操作，如：数据清洗，数据转换，数据抽取等，在此不再赘述。不同的任务可以并行执行，但是一个任务中的资源必须是顺序执行的。例如：任务 T1 一任务 T2，如图 1:



（任务 T1）

（任务 T2）

图 1　任务资源序列图

在资源队列里每个任务的资源并不一定是相邻的，这是因为我们在任务调度的时候也是采用了并发的机制。所以上图 1 中的两个任务 T1 和 T2 中的八个资源的顺序是不确定的 ,每个任务所包含的资源在资源队列中的顺序也可能乱序的 。但是在整个任务的执行过程中每个任务中的这些资源必须是按照任务中规定的顺序来执行[3]。

1.3 传统的资源调度机制

在数据交换的任务中，资源的种类往往是确定的几种或者是几十种，用户在使用交换平台的时候可以根据不同的需求设计不同的任务从而选择不同的资源组合成我们要完成的任务，因此软件的设计者在设计资源调度的时候一般的模型就如图 2 所示：

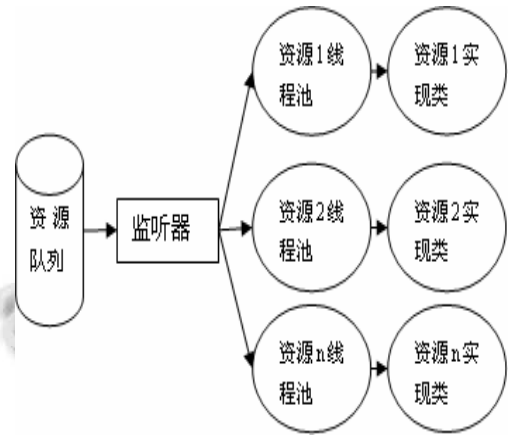


图 2　传统调度模型

在以往的调度模型中，为每一种资源开辟一个线程池，每一个线程所要完成的任务是既定的。在做资源调度的时候由资源监听器监听资源队列中的就绪资源，然后从资源所对应的线程池中取一个空闲线程给资源。如果相应的资源线程池中没有空闲的线程，或者资源的前驱资源没有执行完，当前的就绪资源就会进入阻塞状态，放入资源队列的队尾。

以往的调度策略逻辑很清楚，但是也存在着很多的缺陷：

(1) 影响执行的效率：首先由于系统的原因我们不能设置任意多个线程。因为资源调度和任务调度都是采用多线程并发的机制，所以在大部分时候任务都是扎堆就绪的，所以在资源队列里面会有很多的任务资源一起到达，而大部分任务的资源执行顺序是相同或者是相似的。因此，在资源队列的头部若干个资源一般是任务的第一个资源，很多任务的第一个资源基本上是相同的，一般都是数据加密，这个时候数据加密线程池中的空闲线程数量经常是不足以满足让所有的加密资源都一起执行。这个时候就会有一部分加密资源阻塞，但是其他线程池中却有很多空闲线程得不到利用（其他的线程池中的线程只能执行加密资源的后继操作）加密若没有执行，任务中的所有其他的就绪资源必须阻塞，这样的话很明显既浪费了线程又影响了速率。出现了就绪的资源不能执行，空闲线程在游

荡的情况。

(2) 可维护性和扩展性很差：在这个调度的模型中，如果系统要增加一个新的资源，程序员必须去重新编写一个线程池类和一个资源的执行类，最重要的是我们还要修改资源的监控类等等的其他模块。这样的系统不满足软件设计的开—闭原则，对于以后的维护和扩展带来很大的困难。

2 基于代理模式的资源调度机制的设计

2.1 代理模式

代理模式的最重要的优点就是将重要的关系分离，任何请求的发起都变成对代理的调用，这样就能将访问和实现间接化[4]。代理模式就是强调通信的间接化，利用间接通信改善系统的可维护性和可扩展性。代理模式所涉及到的角色一般有：抽象主题角色：真实对象和代理对象共同的接口。代理主题角色：代理主题角色内部含有对真实主题角色的引用，从而可以在任何时候操纵真实主题对象。真实主题角色：定义了代理角色所代表的真实角色[5]。代理模式一般是和工厂模式一起应用，根据请求，代理主题角色通过业务加载工厂调用真实主题对象，代理并不需要知道它调用的是哪个具体的真实角色，只需要调用抽象类中相应的方法即可。而业务加载工厂从配置文件中得到的代理配置信息，利用 Java 反射机制，工厂会自动选择实例化具体真实对象。基本框图如下图 3 所示：

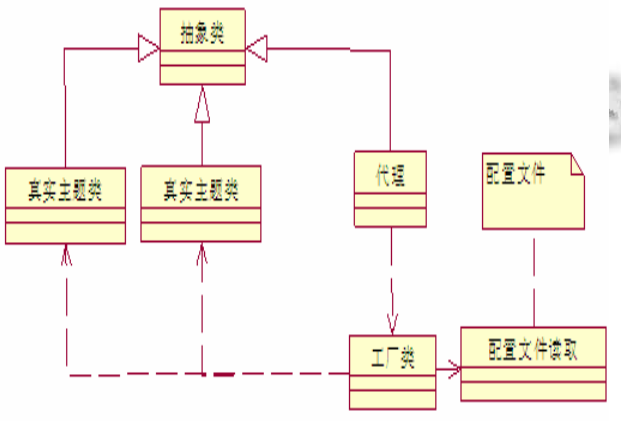


图 3 代理模式和工厂模式类图

2.2 代理模式在资源调度中的应用

在资源的调度中，我们可以把资源队列看做动作的请求者，把线程池看做是动作反馈信息的接受者，因此在资源调度的模型中，应用代理模式之后其基本的框图如下图 4 所示：

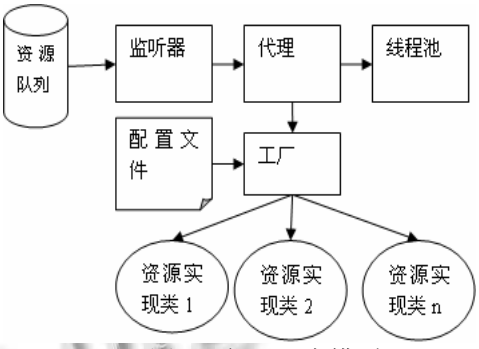


图 4 代理调度模型

如上图所示资源的监听器从资源队列中监听到有关就绪资源的情况就申请代理为它分配一个相应的线程，(这个时候资源监听器已经把资源的信息如资源的数据和资源的种类等等传给了代理）代理把监听器传过来的资源种类的信息传递给工厂，工厂根据这些有关的信息调用配置文件的读写模块，配置文件读写模块根据这些信息去查找配置文件，找到相关的类信息，再反馈给工厂，根据这些信息，工厂就利用 JAVA 的反射机制创建相关的对象，这个对象就是线程要真正执行的资源对象。

2.3 代理模式在资源调度中的实现

代理模式的思想已经应用在我们的数据交换平台的资源调度的模块中，下面介绍这个调度中的五个主要模块。

(1) 代理类：Proxy

图 5 这个类继承了抽象的主题角色 Abstrat Resource, 并实现了接口 Runnble,其中抽象主题角色这个类是描述了，资源类和代理类中最主要的数据和方法，Runnable 是一个线程接口。AbstratResource 中有一个方法 Resourcedo(),这是一个资源要完成的操作，还有一个数据元素 data 这是资源执行过程中用到的数据。在代理类中有两个最重要的方法，一个是 ProxyVoctor(),这是方法实际是得到请求消息的一个方法，另一个是 SendmesToRe-alOb(),这是一个将结果反馈的方法，一般的实现方法就是调用文件读写模块而实现的真正的资源对象反馈给线程池。第二个和第三个方法就是对资源队列和线程池状态的一个记录。

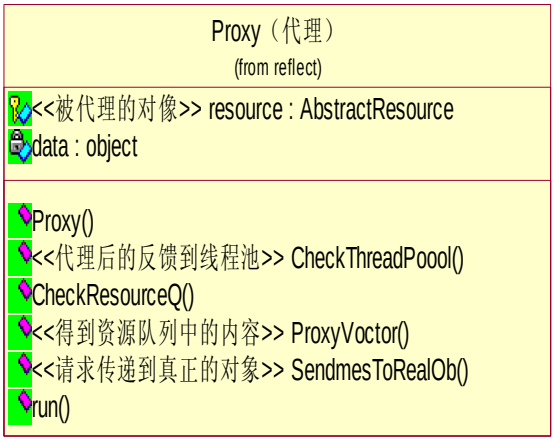


图 5　代理类图

(2) 工厂类：ResourceCreater

这个工厂类是一个利用 JAVA 反射机制自动实例化对象的类。在这个类中有一个重要的数据对象和两个重要的方法。数据对象 resource:Abstrat Resource,这个是抽象资源类的对象。方法：GetClass Name()这个方法是从配置文件中的到具体资源类得完整路径和方法名。方法：CreatResource()这个方法根据资源类得完整路径和方法名在执行期间动态的创建资源的具体对象。

(3) 配置文件读取类：IOPropertiesReader

在本系统中配置文件采用的是.Properties 的格式。在配置文件的读取类中有两个重要的方法，方法一 getProValue()从配置文件中得到类名和完整的路径。方法二 storeProValue()将类名和完整路径名存储到配置文件中。

(4) 线程池：ThreadPool

数据元素：线程池的最大线程数 maxSize,最小线程数 minSize,当前线程数 currentSize,可用线程数 availableSize,这些私有的整数型数据有 get,set 方法读取和赋值。两个重要的方法。AddTask(),这个方法负责给资源分配一个可用的线程，run()这个方法是实现了 Runnable 接口中的方法，是资源具体要实现的动作。

(5) 资源队列监听模块：MonitorPackage

由于这个模块有很多类，实现很多功能，非常的复杂在此就不赘述。

最后给出整个调度模块的基本类图如图 6 所示：

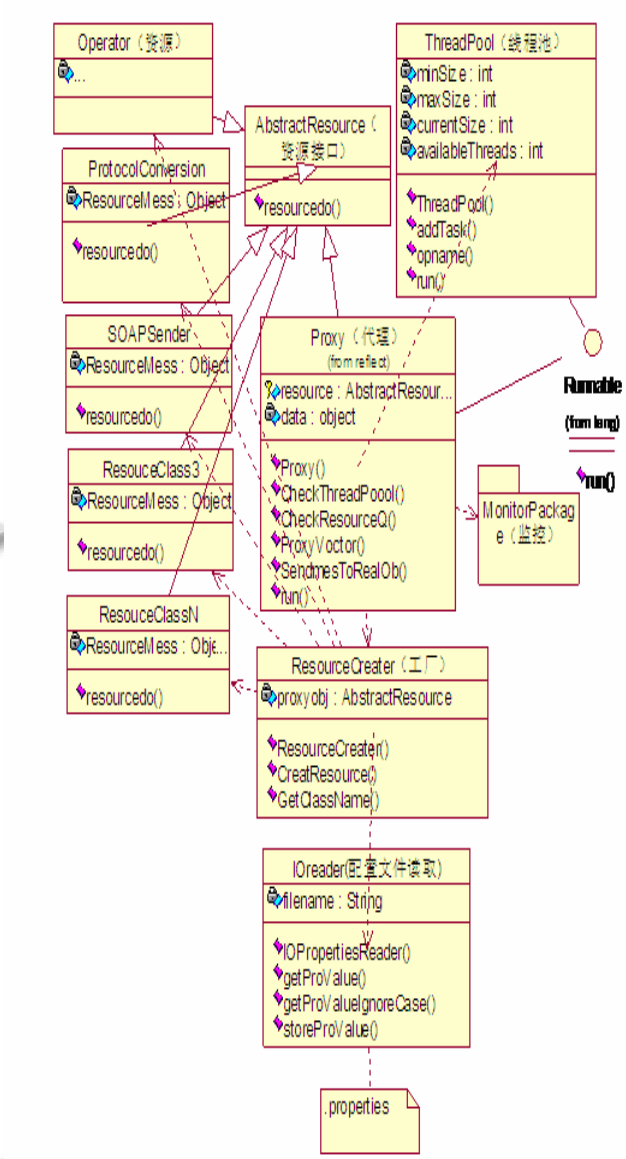


图 6　资源调度类图

3　代理模式应用在资源调度中的优势

在数据交换的资源调度中，由于任务本身的限制（即每个任务中的资源必须是顺序执行的）和资源并发执行的要求，单纯的软件设计给调度模块的维护扩展，甚至是执行效率带来的很多的挑战和麻烦。代理模式和工厂模式的应用就很好的解决了上述问题[6]。

3.1 资源执行效率的提高

代理模式和工厂模式应用之后将会使线程并发的程度提高，从而提高了整个系统的执行效率。

在资源的调度过程中，资源队列中的任务量一般情况下是很多的，资源就绪的数量会多于本资源线程

池中的可用线程数，在旧的调度机制中，就会造成阻塞。在新的调度机制中，只有一个线程池，每一个可用的线程可以执行任何一个就绪的资源这样的话，只要有可用的线程和就绪的资源，那么这个资源就可以执行，不会出现有空闲线程就绪资源却得不到执行的情况。例如：在旧的调度机制中，我们有 20 个空闲线程，其中有 5 数据加密线程，5 个数据抽取线程，5 个数据清洗线程，5 个数据压缩线程。在一次资源调度中有 20 个任务同时到达，他们的第一个资源全都是数据加密，这样只有先到的五个数据加密资源执行，其他的 15 个加密资源因为没有空闲的加密线程而被甩到了队尾，这样的话我们还剩下 15 个空闲线程却还有很多的资源就绪却得不到执行。而在新的调度机制中不会出现这种问题，只要有资源就绪，和空闲线程，资源就能得到执行。

3.2 系统的可维护性和可扩展性提高

在代理模式下的资源调度系统中，如果系统有新的资源出现，我们不需要修改任何代码，只要增加一个我们需要的资源类[7]，并且这个资源类是继承自抽象主题对象 AbstractResource，然后再在资源类的配置文件中加上这个类的整路径就可以了，程序员不必修改任何其他的代码。

4 结语

一个好的资源调度系统既要兼顾执行速率又要考虑以后系统的扩展。设计模式是为了满足软件设计的 OCP 原则（软件开-闭原则）而提出的一种编程思想。用好设计模式并不是一件容易的事，需要丰富的经验和过硬的技术。用好设计模式会给软件的后期维护和扩展带来事半功倍的作用，在本文中设计模式不但提高了调度系统的可维护性和可扩展性，同时也在一定的程度上提高了调度系统的执行效率。这个系统正在调试运行的过程中，还有很多需要改进和升级的地方。

参考文献

1 孙咏.基于 OCP 软件应用架构的设计与实现.北京:中国科学院研究生院,2009.

2 阎宏.JAVA 与模式.北京:电子工业出版社,2002.

3 史捷.数据仓库系统中任务调度策略研究.沈阳:东北大学,2005.

4 Freeman. Head First 设计模式(中文版).北京:中国电力出版社,2007.

5 刘峰.设计模式及组件技术在业务逻辑层中的应用研究.北京:中国科学院研究生院,2010.

6 刘静.模型驱动架构中模型构造与集成策略.上海:华东师范大学,2006.

7 Wang XB. Research and Implementation of Design Pattern(中文版). Oriented Model Transformation. Computer Society: 2008.

(上接第 222 页)

较小；（2）在图像上对建筑物可见部分的轮廓进行简单的提取。建筑物越复杂,拓扑结构提取消耗的时间就越多,整个匹配效率也就低。图像的质量则关系到能否提取到完整、准确的边,这对匹配的稳健性有很大的影响；（3）图像中的建筑物与模型物体匹配。这个阶段使用的匹配算法是一个关键问题，它的好坏对整个系统的运行效率影响很大。

本文采取的方法简单、高效，它只需对可见部分的拓扑结构进行分析，并不需要完备的拓扑信息，这对于有遮挡的建筑物的重建很有帮助；同时，它跨越了复杂的数学运算，能够节省匹配的时间。

参考文献

1 夏春林,王佳奇.3DGIS 中建筑物三维建模技术综述.测绘科学,2011,36(1):70−72.

2 Ting Z, Feng DD, Zheng T. 3D Reconstruction of single picture. ACM Conference on Visual Information Processing. 2004. 83–86.

3 吴军.三维城市建模中的建筑墙面纹理快速重建研究.测绘学报,2005,34(4):21−24.

4 张会霞,陈宜金,刘国波.基于三维激光扫描仪的校园建筑物建模研究.测绘工程,2010,19(1):32−35.

5 Van den Heuvel, FA. Reconstruction from a single architectural image from the Meydenbauer archives. Proc. of the 18th International Symposium of CIPA 2001. Potsdam, 699.

6 吴培景,陈光梦.一种改进的 SSDA 图像匹配算法.计算机工程与应用,2005,33(11):75−78.