第16卷 第8期
2006年8月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.16 No.8
Aug. 2006

基于JMS的企业消息系统的设计与实现

潘 涛,张能立
(武汉理工大学 计算机科学与技术学院,湖北 武汉 430070)

摘 要:文中首先介绍了Java消息服务的技术特点,讨论了消息中间件的基本设计思想,并论述了运用JMS技术为某钢铁交易市场设计的一个实时报价平台的实现流程。该平台采用分布式框架构建,可以同时支持瘦客户端和胖客户端。

关键词:Java消息服务;消息中间件;消息驱动Bean
中图分类号:TP311.1 **文献标识码**:A **文章编号**:1673-629X(2006)08-0149-03

Design and Implementation of Enterprise Message System Based on JMS

PAN Tao, ZHANG Neng-li
(School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China)

Abstract: Firstly, introduced some technologies about Java message service, and discussed the design thought of message-oriented middleware. Then it expatiated on the implementation flow of a real time quote system for one steel exchange market using JMS technology and distributed framework. It can support thin client side and fat client side at the same time.

Key words: Java message service; message-oriented middleware; message driven bean

0 引 言

当前企业信息化程度逐渐提高,企业内部的应用系统日益复杂;构建各个系统所涉及的数据结构、操作系统、通信协议、数据库和其他相关服务各不相同。为了信息共享,这时企业需要一套通讯系统来连接各个子系统,消息中间件可以用来解决这个问题。它能在不同的系统平台中提供稳定、可靠的消息传输。Sun公司EJB2.0中定义了相关的Java消息服务规范,提供了一组公用的接口,其他的中间件厂商则负责具体的实现,提供相关的Java中间件产品,如JMS容器,及一些扩展的接口等。文中侧重介绍Java消息中间件的相关技术及其应用。

1 Java消息服务

Java消息服务对于网络通信而言是一个轻量级的通信工具,相对于传统的、重量级的通信工具RMI-IIOP而言,它有许多的优点,如:

1) 消息客户端在请求远程响应时,不必阻塞调用。当Client提出请求后,马上提供其它的服务,不必如RMI-IIOP调用一样阻塞在那里,等待远程系统响应。

2) 消息服务同时也可以提供对N元通信的支持。消息中间件产品可以接收多个发送者的消息,然后将这些消息广播到多个使用者[1]。

Java消息机制主要有两种类型的域:发布/订阅方式和点到点方式。

它支持的消息类型有:流消息(如字节流等),MapMessage(名称/值对消息),对象消息(该对象必须实现序列化接口),TextMessage(字符串)。

下面简述Java消息服务系统的编程基本流程:

(1) 使用JNDI定位Java消息服务驱动程序;

(2) 使用连接工厂创建Java消息服务连接;

(3) 创建Java消息服务会话;

(4) 定位Java消息服务的目的地;

(5) 创建Java消息服务发生者或者Java消息服务的使用者;

(6) 发送或者接收消息。

这里给出一个可序列化的Java对象,采用发布/定阅的方式通过JMS把该对象的一个实例序列化后,从客户端传送到使用者端,订阅者再把该对象反序列化进行使用。

在这里以BEA Weblogic JMS产品作为例子[2]。

编程的模型如图1所示。

● 需要传递的Java对象:

```
class JmsData implements Serializable
{
  private string field0;
```

收稿日期:2005-11-29

作者简介:潘 涛(1980-),男,湖北洪湖人,硕士研究生,研究方向为网络数据库及中间件技术;张能立,副教授,硕士生导师,研究方向为网络数据库及中间件技术。

```
  private string field1;
  public JmsData(string s0,string s1) {this. field0 = s0; this. field1
= s1;}
}
```

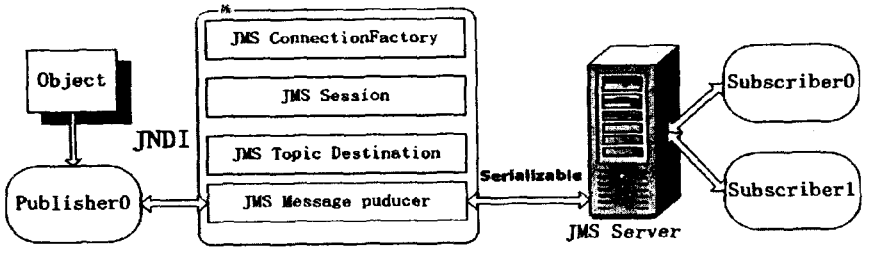

图 1 Java 主题消息服务流程图

● 主题发送者主要的接口代码:

```
// 导入必要的类包,如 javax. naming. * ;javax. jms. * 等
class JmsClient
{
  public static void main(String[] args) throws Exception
  {
    //初始化 JNDI 服务;
    Hashtable env = new Hashtable();
    env. put(Context. INITIAL_ CONTEXT_ FACTORY,JNDI_
FACTORY);
     env. put (Context. PROVIDER_ URL, "t3://localhost:
7001");
    env. put("weblogic. jndi. createIntermediateContexts","true");
    Context ctx = new InitialContext(env);
    //通过 JNDI 查找连接工厂;
    TopicConnectionFactory tconFactory =
    (TopicConnectionFactory)PortableRemoteObject. narrow(
     ctx. lookup (JMS-FACTORY), TopicConnectionFactory.
class);
    //使用连接工厂创建 JMS 连接;
    TopicConnection tcon = tconfactory. createTopicConnection();
    //使用连接创建会话;
    TopicSession tsession =
    tcon. createTopicSession(false, Session. AUTO_ ACKNOWL-
EDGE);
    //通过 JNDI 查询主题目的地;
    Topic topic =
    (Topic) PortableRemoteObject. narrow (ctx. lookup (topic-
Name),Topic. class);
    //创建消息发布者;
    TopicPublisher tpublisher = tsession. createPublisher(topic);
    //创建对象消息
    JmsData aInstance = JmsData("abc","def");
    ObjectMessage om = tsession. createObjectMessage();
    om. setObject(aInstance);
    //发送消息
    tpublisher. publish(om);}}
```

● 主题订阅者主要的接口代码:

```
public class TopicReceive implements MessageListener
{
  public void init()
  {//同前发送者设置 JNDI 连接工厂,创建连接,创建会话,
查找主题等……}
        public void onMessage(ObjectMessage
omTemp)
        {
          JmsData bInstance;
            if (omTemp instanceof Ob-
jectMessage)
            {bInstance = ((ObjectMessage)
omTemp). getObject();}
      else {//其他处理}
      //得到了反序列化的对象实例,即可进行相应的业务
逻辑操作了……
    }
    public static void main(String[] args) throws Exception
    {
      TopicReceive tr = new TopicReceive();
      tr. init();
      synchronized(tr)
      {
        while (! tr. quit)
        {try {tr. wait();} catch (InterruptedException ie) { e.
printStackTrace();} }
      }
    }
    //其他方法,如关闭连接等
}
```

2 Java 消息中间件

Sun 公司提供了一套统一的 JMS 接口,各个中间件厂商可以运用不同的技术去实现这套接口,成为各具特点的 JMS Provider。不过,因为它们实现的是同样一套接口,所以相同的 JMS 消息生产者代码和 JMS 消息消费者代码可以很方便地移植到不同的中间件产品上运行。这也是 Sun JMS 团队的设计初衷[3]。

Java 消息中间件虽然可以有不同的实现方法,不过还是必须遵循一些共同的特点:

(1)通讯层的相关组件必须用纯 Java 代码编写。因为 Provider 端对 JMS 接口的实现,如 TopicConnectionFactory,Topic 等必须通过 Java 的序列化接口传递到客户端。不过,本地持久化相关的类,如 I/O 操作可以部分用其他语言如 C 语言来进行实现,然后通过 JNI 通道与持久化类进行连接。

(2)中间件产品起码要提供 3 种类型数据包的实现,即消息包、控制信息包、确认回执包。消息包显然是传递客户端的消息,可以在 Provider 端定义一个 ImessageDatagram 接口来定义一套标准的方法用于与客户端的通信;控制信息包是用来传递相关的控制信息的,如订阅、取消

订阅、启动会话、停止会话等,同样可以定义一个 Icontrol-Datagram 接口。确认回执包是用来为上面两个包提供确认收到或延时丢失等服务的,可以定义一个 IAckDatagram 接口[4]。

(3)作为企业级的应用,该产品同时应该提供相关的事务处理和纠错的机制。如消息排序的问题,保证时间上领先的消息先被消费者处理。另外,如果出现异常情况,如网络传输等,生产者没有收到确认回执包,所以它继续生产同样的消息提供给消费者,这个消息实际上就是"脏消息",中间件产品的设计必须妥善处理该类问题。

(4)消息中间件应该被设计为可插入式使用组件。EJB2.0 定义了一套新的 Bean 组件,消息驱动 Bean 来作为 JMS 中间件的消费者,所以消息中间件产品可以和 EJB 服务器集成到一起,共享一个 JVM,提高性能,并作为一个整体的产品提供给用户[5]。

目前,业内已经有数家公司提供 JMS 中间件产品,如 BEA WEBLOGIC JMS, IBM WEBSPERE JMS 等。如果企业需要多次使用该产品,为了节省投资,也可以自己实现一个 JMS 中间件产品,进行重复使用。下面给出一个具体的企业应用,来简述企业使用 JMS 的流程。

3　Java 消息服务的一个具体应用

(1)企业需求。

某钢铁交易市场经营着数十种不同类型和规格的钢材。为提高交易的效率,企业需提供一套网络报价系统给 Internet 客户来查询当天钢材的交易信息,如钢材的价格、规格、成交量、库存量、总成交额等。

该系统针对不同的客户群体,可以提供两种不同类型的客户端:

①胖客户端。需要安装相关的客户端软件,类似股票报价系统,响应速度快,延迟在数秒之内,一般供专业性人士使用。

②瘦客户端。客户使用 IE 连接到固定 IP 的站点,采取页面自动刷新或 Java Applet 定时更新的方式,查看实时的信息。但信息的更新较胖客户端稍慢,不过延迟也可控制在很短的时间之内,但是客户使用方便,不需要安装特定的客户端软件。

(2) 系统设计。

系统设计如图 2 所示,数据发布服务器作为消息的生产者,它一般由 3 个模块组成:

①数据提取模块。该模块和数据库保持常连接,定时循环查询数据库,得到所有在卖钢材的信息,封装成一个 Java 对象,传递给数据分析层进行处理。

②数据分析层模块。该模块得到数据后马上和内存池中的数据进行比较,如有变化则更新内存池,并将变化的钢材个品或全部钢材数据传递给消息发布层;

③消息发布层。得到变化的数据后,先判断该发送怎样的数据包。可以定义如下两种数据包:

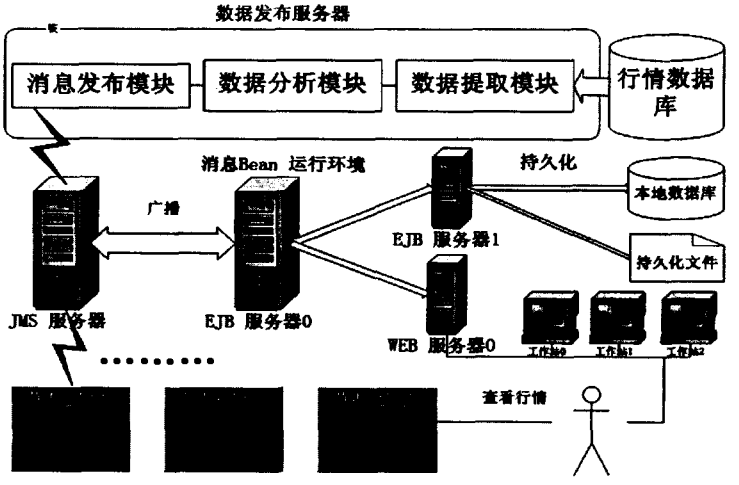

图 2　行情系统结构设计图

* 全部的钢材初始数据包,可以采用定时发送的方式;

* 部分发生变化的钢材数据包,可以采用实时发送的方式。

判断完成后传递给 XML 包装逻辑部分,把数据包装成 XML 的格式发布给 JMS 中间件。

JMS 中间件得到发布的数据后,利用缓冲机制在服务器上缓冲一下,搜索在线的订阅客户,如:同一个容器的消息驱动 Bean, Internet 上的胖客户端等,得到一个待发送列表,把数据分发出去。

消息驱动 Bean 在得到消息后,触发消息处理逻辑,对内存池中的钢材数据进行更新,并持久化到文件或本地数据库。节点上的其他的 EJB 组件,如:Session Bean, Entity Bean ,取出更新后的数据,进行持久化操作或响应 Web 层的请求,如 Servlet 等,然后通过 Web 层将数据广播到各个瘦客户端。

以下是相关的接口代码:

```
public class JmsBean implements MessageDrivenBean, MessageLis-
tenser
{
    MessageDrivenContext messageDrivenContext;
     public void setMessageDrivenContext (MessageDrivenContext
messageDrivenContext)
    { this. messageDrivenContext = messageDrivenContext; }
    //其他一些 Message - Driven Bean 必须实现的方法,如 e-
jbCreate(),ejbRemove()等
    public void onMessage(Message msg)
    {
      TextMessage tm = (TextMessage)msg;
      try{ String text= tm. getText();}
      catch(JMSException e) { e. printStackTrace() ; }
```

(下转第 154 页)

就大大减少其中的数据交互,这点对于带宽资源较为紧张、网络拓扑复杂的网络至关重要[5]。

(3)利用 Mobile Agent 提高信息的轮询速度,减小网络管理负担。

网管对网络中所有设备的定时查询是采用轮询的方式(polling),由网管在一段时间间隔对每台设备进行 polling 操作,这样必然导致两个问题:一是在网络节点数较多的情况下,网管的工作负担过重,限制了网络管理的规模;二是由于节点数的增加,网管轮询一遍网络的时间必然增大,这就限制了对轮询间隔大小的选择,使管理信息的实时性较差,尤其是在目前网络带宽不断增长的情况下,网络设备的数据流动量很大,设备的状态在很短的时间内会有很大的变化,实时性问题便更为突出。

利用 Mobile Agent 技术可以有效地解决这两个问题,生成多个拓扑代理,同时向网络的不同节点派发,由代理在远端执行网管操作,将过去由网管统一进行的收集、处理工作交给在网络不同部分的各个代理进行,而网管只负责对最后信息的接收,由于网管工作范围由一个网络变成几台网络设备或局部网络,运行的效率提高,同时多个代理的并发执行也大大提高了信息的检索速度,使信息的轮询间隔不受节点数增加的限制。

(4)配置网络设备更加灵活。

传统的分布式网络管理在服务器上描述服务被执行的代码和客户端上描述怎样请求远程服务的代码是与主机静态绑定的,当所需要的信息超出 MIB 定义的范围时,就需要对现有的网络设备系统软件进行更新,这就带来了很大的不便。在 Mobile Agent 的管理模式中,可以突破原有的 MIB 定义的限制,根据需要,在网络异常情况下定义相应动作,从而提高配置网络设备的灵活性。

4 结束语

基于移动代理的网络管理方案很好地解决了当前集中式网络管理所带来的问题,将原本完全或大部分由网管站承担的管理计算任务分布到网络各节点上,从而减轻了网管站的计算负载,减少了网络管理对带宽的要求,同时提高网络管理功能的灵活性和可重构性,适应管理功能的发展和变化,尤其适合于当今在地理上越来越分布的网络环境,具有很好的实用和研究价值。

但是针对移动代理系统的安全性方面所做的研究还远远不够,如集中在对移动代理的合法性验证、对移动代理所携带的数据的保护以及防止某些恶意攻击及对执行环境的非法修改等。另外,对于地理上分布的多个有层次的域进行管理时,各域的网管站之间、同域中各移动代理之间的分工协作和信息交换也需要进行更深入的探讨。

参考文献:

[1] 郭 军. 网络管理[M]. 北京: 北京邮电大学出版社, 2001.

[2] 陈 萍,许卓群,刘贺湘,等. 基于智能移动代理技术的网络管理系统综述[J]. 计算机应用研究, 2003(3):1-3.

[3] 张云勇,刘锦德.移动 Agent 技术[M].北京:清华大学出版社,2003.

[4] 赖秀金,王 乘. Mobile agent 在网络管理中应用[J].微机发展,2004,14(9):10-13.

[5] 张普含,孙玉芳.一种基于移动代理的网络管理系统及性能分析[J].软件学报, 2002,13(11):90-98.

(上接第 151 页)

```
//得到 xml 数据后,可以运用相关的 xml 解析器把数据转化成别的格式
}
//进行其他一些相关处理,如内存数据更新,持久化到文件系统等……}
```

同时,JMS 胖客户端得到数据后马上调用 onMessage()方法对本地数据进行更新,并持久化。客户端 GUI 程序把更新后的数据马上显示出来。

4 结束语

Java 消息服务规范作为 Sun J2EE 标准的有机组成部分,具有 J2EE 组件共有的优点,如跨平台部署、互操作性良好、众多厂商的支持等。成熟的中间件厂商已经把 Web 容器,EJB 容器,JMS 容器集成一个整体,共享一个虚拟机进行运行。因此,JMS 同时拥有容器提供的事务处理、集群部署、数据库连接池等诸多特点。另外,相对其他组件技术而言,它是个轻量级、非阻塞式调用的组件,拥有不错的灵活性。综上所述,JMS 技术能为企业消息系统的设计提供不错的解决方案,但它也不是可以解决任何问题,如在解决异构系统的通信问题上,WebService,CORBA 等技术则更适合一些。

参考文献:

[1] Roman E. Mastering Enterprise JavaBean(Second Edition)[M].刘晓华,等译. 北京:电子工业出版社,2002.

[2] Bea. Weblogic JMS Documentation[EB/OL]. http://www.bea.com,2002.

[3] Sun. Java Message Service Specification[EB/OL]. http://www.sun.com,2002.

[4] Rhombus Technologies, Inc. UberMQ JMS Documentation [EB/OL]. http://www.ubermq,2003.

[5] Shoffner M. Write your own MOM[EB/OL]. http://www.javaworld.com/javaworld/jw-05-1998/jw-05-step.html,2003.