# 基于 ActiveMQ 的异步消息总线的设计与实现①

戴 俊 朱晓民 (北京邮电大学 网络与交换技术国家重点实验室 北京 100876;
东信北邮信息技术有限公司 北京 100191)

摘 要: MOM (Message-Oriented Middleware, 消息中间件) 是解决异构分布式系统中通信和排队问题的中间件技术, ActiveMQ 则是 MOM 的一个跨语言跨平台实现。首先介绍了 ActiveMQ 的特点, 并对其性能和稳定性进行了对比测试。然后, 在其基础之上增加了消息总线控制功能, 构建了一个通用的异步消息总线。经过实践证明, 该消息总线可以提高移动增值业务的开发效率和处理性能。

关键词: MOM;JMS;ActiveMQ;消息总线

## Design and Implementation of an Asynchronous Message Bus Based on ActiveMQ

DAI Jun, ZHU Xiao-Min (State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China; Eastcom-BUPT Information Technology Co. Ltd., Beijing 100191, China)

**Abstract**: Message-Oriented Middleware (MOM) is an enterprise middleware technology, which is designed to solve the communication and queuing problems in heterogeneous distributed system. ActiveMQ is a cross-platform and cross-language implement ation of MOM. Firstly, this paper introduces its features and makes a back-to-back testing in stability and performance. Then, a bus controller module is added to it and the common asynchronous message bus is built. It is proved that the message bus can improve the development efficiency and processing performance of mobile value-added business.

**Keywords**: MOM;JMS; ActiveMQ; message bus

## 1 引言

随着网络技术飞速发展, 网络的带宽和质量得到了极大地提高, 独立部署的系统通过网络建立了彼此间的联系[1]。再加上企业应用对网络融合、数据挖掘、经营分析的需要, 也促使着独立系统之间的交互。于是, 企业的计算环境就从传统的集中式变成了分布式。在分布式环境下, 系统的异构性, 网络的带宽, 连接的稳定性最终影响着服务质量。为满足在分布式计算环境下企业应用对性能、安全性、稳定性等方面的要求, 可以构建基于消息中间件 (Message-Oriented Middleware, MOM) 的数据通信系统。它能够异步传递消息, 将彼此独立的计算机连接起来组成松耦合的系统, 并且可以有效地屏蔽异构细节对外提供统一服务。

作为解决异构系统通信和排队问题的有效手段, MOM 自诞生以来就成为了企业应用的关键组件。近年来, MOM 在企业中得到了广泛的研究和应用, IBM、微软等软件厂商纷纷实现了自己的 MOM 产品。但是,

① 基金项目:国家杰出青年科学基金(60525110);国家重点基础研究发展规划(973)(2007CB307100,2007CB307103);国家自然科学基金(60902051);中央高校基本科研业务费资助(BUPT2009RC0505);电子信息产业发展基金项目
收稿日期:2009-12-13;收到修改稿的时间:2010-02-15

由于没有统一的标准，不同厂商的 MOM 产品很难实现无缝的交互操作，这反而给整个计算环境增加了新的异构性和复杂性。为解决这一问题，JCP（Java Community Process, Java 标准组织）于 2003 年发布了 Java 领域的 MOM 规范 JSR914[2]，即 JMS（Java Message Service, Java 消息服务）。JMS 不负责 MOM 的实现，只定义了 MOM 系统的服务和接口，包括点对点和订阅/发布两种模型，可以提供可靠的消息传输、事务和过滤等服务。随着 Java 技术在企业领域得到广泛应用，JMS 逐渐成为了企业领域事实上的 MOM 标准，诞生了一批以跨平台跨语言消息中间件为目的基于 JMS 的 MOM 实现，例如 ActiveMQ[3]。

## 2 ActiveMQ系统

### 2.1 ActiveMQ 简介

ActiveMQ 是一款由 Apache 组织发布的开源的(Apache License 2.0)企业级通用消息中间件，完整实现了 JMS1.1 和 J2EE 1.4 中 JMS 服务，还具备了集群、事务、存储转发、持久化等企业特性。自从 2006 年诞生以来，ActiveMQ 已经发展到 5.3 版本，被越来越多的企业采用，甚至直接捆绑在自己产品中，是目前最流行的企业级消息中间件。

ActiveMQ 是一款独立的 MQM 产品，借助于 JVM 跨平台的特性，ActiveMQ 可以完美地运行于所有主流操作系统中。虽然它是由 Java 语言编写的，但是这并不意味着 ActiveMQ 完全依赖于 Java，只能用于 Java 程序。实际上，ActiveMQ 实现的只是 JMS 规范中的消息服务提供者，任何符合 JMS 规范编写的应用程序，都能通过网络使用其消息队列服务。ActiveMQ 提供了丰富连接方式，按照耦合的紧密程度可以分为四种：直接内嵌于 JBoss 等 J2EE（Java 2 Enterprise Edition, Java2 企业版）容器，直接利用 AMOP、Openwire 等标准协议，通过适配器与 Java、C++、AJAX 等程序连接以及桥式连接其它 JMS。

### 2.2 ActiveMQ 特点分析

ActiveMQ 是 Java 语言编写的基于 JMS 消息中间件，借助于 Java 程序“一次编写，多次运行”的特性，可以完美地运行于 Linux，Windows，Unix 等操作系统。从图 1 中可以看出，ActiveMQ 内部实现采用的是分层和插件相结合的系统架构，从内到外一共可以分为三层，分别是 JMS 核心层、接口层和插件层。前两层提供了基本的 JMS 功能和接口，而第三层则由不同的插件构成向外提供队列、集群等服务。

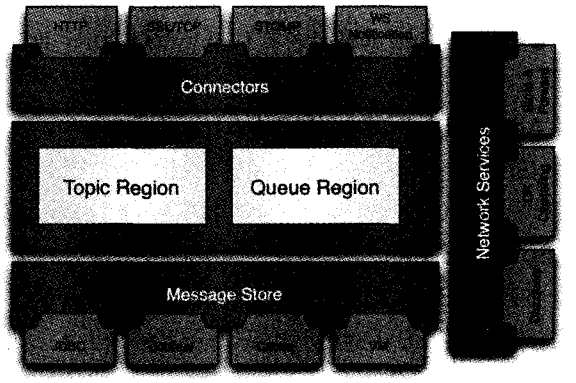


图 1　ActiveMQ 架构图

JMS 核心层是 JMS 服务器端的实现，完整支持 JMS1.1 和 J2EE 1.4 规范，可以部署到任何兼容 J2EE 1.4(及以上)的 J2EE 容器上(如 JBoss)。接口层根据功能的不同又可以分为三个子模块，分别是通信接口、消息保存接口和网络服务接口子模块。

通信接口负责网络连接和消息传输，通过分布式命令模式解除了消息内容和通信载体的紧耦合关系，以便于消息在网络中进行传输。它与各种协议插件一起完成消息在 JMS 客户端与服务器端和不同 JMS 之间的传输。由于插件应用层次的不同，ActiveMQ 拥有了三种自底向上的网络通信能力：

- TCP，UDP，SSL，NIO，JATX 等底层网络传输
- OpenWire, Stomp REST, WS Notification, XMPP，AMQP 等标准协议链路
- Java，C，C++，C#，Ruby，Perl，Python，PHP，AJAX 等语言级交互通道

通过消息保存接口 ActiveMQ 可以支持内存、文件、内嵌数据库和外部数据库等四种消息持久化方式。网络服务接口是高级功能的接口，支持存储转发、集群、命名服务等。根据应用场景的不同，用户可以自由选择不同的插件组合。ActiveMQ 以企业应用为目的，在设计上就保证了高性能的集群，目前支持客户端-服务器端和点对点两种扩展方式。

此外 ActiveMQ 还拥有了以下的特性：

- 具备 JMX 和 WEB 两种控制台，便于开发调试和运行监控
- 当以内存式 JMS 运行时，可与单元测试无缝集成
- 所有功能和特性都可以通过 XML 文件进行配置

### 2.3 ActiveMQ 与 JBossMQ 的对比测试

JBoss 是一款企业级中间件产品，被广泛用于企业领域，例如作为短信增值业务执行容器。JBossMQ 是 JBoss 内置 JMS 服务实现，与 JBoss 共享内存等资源，依赖于 JBoss 提供的 J2EE 环境。而 ActiveMQ 既可以部署于 J2EE 容器内又能部署于容器外。当它通过进程形式独立提供消息队列服务时，即使业务环境失效也不会导致到整个系统崩溃。根据短信增值业务的特点，对这两款 JMS 产品在稳定性和性能上进行了对比测试，测试环境如表 1 所示：

表 1　测试环境

| | |
|---|---|
| 主机 | HP DL 380 G4 378735-AA1<br>Intel Xeon 1.60GHz X 2，8 核 / 4G 内存 |
| 操作系统 | RedHat Enterprise Advanced Server 4.0 |
| 数据库 | Oracle 10g |
| J2EE 容器 | JBoss 4.3 EAP |
| JMS 连接数 | 20 |
| 参数设置 | 每条消息的长度是 2KB<br>不开启内存以外的消息持久化方式 |

测试结果如表 2 所示：

表 2　测试结果对比

| 测试对象 / 测试项 | ActiveMQ 5.3 | JBossMQ/JBoss EAP 4.3 |
|---|---|---|
| 最大消息堆积（万条） | 100 | 35 |
| 消息吞吐量（条/秒） | 8000 | 1000 |

经过分析发现，ActiveMQ 的最大消息堆积数是 JBossMQ 的 3.3 倍。在只启用内存持久化的情况下，消息堆积数直接反映可利用的内存容量，最大可用内存容量=消息长度 X 最大消息堆积数。经过计算发现，JBossMQ 作为内嵌式 JMS 需要与 JBoss 共享内存，实际上最大可用内存约为 700MB，远小于 ActiveMQ 可以独占的 2GB 内存空间。在消息吞吐率方面，保证消息零积压要求下，ActiveMQ 每秒可以稳定收发 8000 条消息是 JBossMQ 的 8 倍。另外一方面，ActiveMQ 具有集群等高级企业特性，拥有与非 Java 程序交互的能力有利于保护企业已有的投资。所以，ActiveMQ 比 JBossMQ 拥有更好的稳定性和性能，更能满足短信增值业务等企业应用对消息中间件要求[4]。

## 3　消息总线的设计与实现

### 3.1 架构设计

ActiveMQ 提供了点对点和订阅/发布两个模型的消息队列，前者在端点之间建立了虚拟链路把消息从源地址发送到目的地址，消息的发送必须在接受之后，采用的是同步方式；后者不需要建立端到端的链路，生产者根据主题把消息发送到 ActiveMQ 的不同队列，由 ActiveMQ 负责把队列中的消息发送到对应的主题订阅者，消息接收和发送可以同时进行，是典型的异步方式。从系统的角度上说，消息总线是内部的数据通道，生产者负责产生消息加入总线，由总线完成路由消息到合适的消费者。对于模块而言，在某一时刻可能是消息的生产者（产生输入），在另一个时刻可能作为消费者（响应结果），甚至同时担任两个角色。为了同时保证高吞吐率和双工通信，设计消息总线时选择了主题订阅/发布的异步模型，并为每对 JMS 客户端建立了成对消息队列，如图 2 所示的短信前置和后置队列。

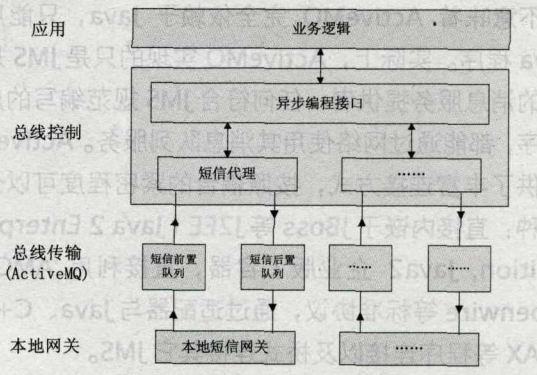


图 2　消息总线的架构

如图 2 所示，消息总线由传输和控制两部分组成，向下连接本地网关提供通信能力，向上提供异步的编程接口为具体应用服务。ActiveMQ 作为基本消息队列以主题订阅/发布模型提供异步的消息服务，再按照

协议和业务等不同主题划分为成对的子队列。所有的子队列分别与控制模块的主题代理和本地网关建立连接，经由编程接口屏蔽实现和构成的细节，最终组成消息总线提供统一的服务。消息总线还为业务逻辑提供了一套易用的基于事件的异步 API（Application Programming Interface，应用程序接口），以提高业务开发的效率并满足高速执行的要求。

### 3.2 可配置的控制模块

总线控制模块主要负责总线初始化、消息收发、流量控制、业务路由等一系列控制功能。该模块部署于 JBoss 内部，初始化工作在 JBoss 启动阶段完成，并且采用了延迟加载技术以解除潜在的部署冲突。ActiveMQ 队列中的消息由主题代理线程进行收发，这些线程运行于控制模块内一系列线程池中。通过调整每个线程池的容量和队列长度，可以控制消息收发速率，还能根据主题的优先级有针对性的进行处理。总线则负责把接收到的消息路由到正确的业务逻辑，以及把业务逻辑产生的消息路由到正确的 ActiveMQ 主题队列。为了适应不同业务场景的需要，控制模块的所有功能和特性都可以通过 XML 进行配置，如下所示：

```
<Bus>
<lazyTime>60</ lazyTime ><!-- 部署延后时间(秒)-->
<Consumers>
<queue>
<url>tcp://10.1.70.138:61616</url><!--ActiveMQ 的地址-->
<queueName>queue/sms_idp</queueName>
<connectTimeout>30</ connectTimeout ><!--超时(毫秒) -->
<reconnectInterval>1000</reconnectInterval><!-间隔-->
<batchSize>2000</batchSize><!--批处理大小-->
<receiveWaitNum>10</receiveWaitNum><!--重试次-->
<logicBeanName><!--后续处理逻辑-->
<value>IdpSmsRecvServiceBean/local</value>
</logicBeanName>
<thread>
<corePoolSize>2</corePoolSize><!--程池中核心线程数->
<maxmumPoolSize>5</maxmumPoolSize>
<maxQueueSize>5</maxQueueSize><!-线程池队列长度->
<connectNum>3</ connectNum ><!-- 连接的线程数 -->
<rate>10000</rate><!--消息最大处理速率(消息/秒) -->
<scheduleWait>30</ scheduleWait ><!--调度等待(毫秒) -->
<keepAliveTime>1000</keepAliveTime><!--空闲时间-->
</thread>
</queue >
</Consumers>
<Producers>
< queue >
<queueName>queue/idp_sms</queueName>
<url>tcp://10.1.70.138:61616</url><!—ActiveMQ URL -->
<threadNum>3</threadNum><!-- 连接的线程数-->
<threadRate>10000</ threadRate ><!—最大速率(秒) -->
<reconnectInterval>1000</reconnectInterval><!-重连-->
<scheduleWait >30</ scheduleWait ><!--调度等待(毫秒)-->
</ queue >
</Producers>
</Bus>
```

### 3.3 异步编程接口

异步编程接口实现了 EDA(Event-Driven Architecture，事件驱动架构)[5]，利用事件订阅/分发方式解除业务逻辑与消息内容的紧耦合关系，使得业务的异步处理成为了可能。它驱动着整个业务处理流程，需要完成消息到事件的封装，事件的订阅到分发等一系列过程。首先代理线程从消息队列中获取消息，按

(下转第 215 页)

Proceedings of SPIT-IEEE Colloquium and International Conference, Mumbai, India. Vol.1, 8 – 13.
3 Land E H, McCann J J. Lightness and Retinex Theory. Journal of the Optical Society of America, 1971, 61(1): 1 – 11.
4 李云,刘学诚.基于小波变换的图像增强算法研究.计算机应用与软件,2008,8(8):100 – 103.
5 朱立新,王平安,夏德深.基于梯度场均衡化的图像对比度增强.计算机辅助设计与图形学学报, 2007, 19(12):1546 – 1552.
6 周俊,罗挺,路翔,王冰.一种基于拉伸变换函数的指纹图像增强算法.计算机系统应用,2009,3(3):141 – 143.
7 Tao L, Seow MJ, Asari VK. Nonlinear Images Enhancement to Improve Face Detection in Complex Lighting Environment. International Journal of Computational Intelligence Research. 2006,2:327 – 336.
8 Yale Face Database.[2009-10-28].http://cvc.yale.edu / projects/yalefaces/yalefaces.html.
9 Sim T, Baker S, Bsat M. The CMU Pose, Illumination, and Expression(PIE) Database. Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition, May, 2002.

(上接第 257 页)

照预先定义的规则封装成事件，再把事件交由调度子模块统一处理。事件调度子模块收到事件后,根据XML配置信息识别出事件的主题，最后发送到对应的后续逻辑进行处理。因为，事件屏蔽了消息的细节，独立进行触发，异步进行处理。在实现后续逻辑时，只需要针对具体事件编程，而不必再考虑消息到达的先后顺序或者逻辑冲突。

## 4 结论

目前，基于该消息总线开发的短信增值业务已经在贵州联通上线运营，开发效率和性能都得到了较大提高。一方面，由于该消息总线解除了业务和通信之间的耦合，我们在业务开发中采用了事件编程模型，缩短了一半的开发时间，在 6 个月内就完成了开发和实施。另一方面，新的短信增值业务系统得益于ActiveMQ高达 8000 消息/秒的消息吞吐率，短信增值业务处理峰值也从 185 业务/秒提高到了 902 业务/秒，提高了约 3.86 倍。

ActiveMQ 作为一款通用消息中间件，在功能、性能和稳定性方面都有着优异的表现。利用它构建消息总线可以屏蔽组件之间的异构性，既能保护企业现有投资又便于快速构建高性能高稳定性的松耦合企业应用。并且经过实践证实，该消息总线可以满足移动增值业务快速开发、稳定高效的要求[6]。我们下一步将研究 ActiveMQ 的集群等高级企业特性，进一步提高该消息总线的扩展性和稳定性。

### 参考文献

1 李晓东.一种智能消息中间件的研究. 黑龙江科技信息, 2009,24:49 – 49.
2 Nigel Deakin. JSR 914: Java™ Message Service (JMS) API. http://jcp.org/en/jsr/detail?id=914. 2003,12.
3 The Apache Fundation. Apache ActiveMQ. http:// activemq.apache.org/. 2006, 4.
4 Zhang Y, Liao JX, Zhang TY, Zhu XM, Ma J. A Novel Method for the Short Message or Multimedia Message Synchronization. Proc. of IEEE International Conference on Wireless and Mobile Communications 2006.(ICWMC2006), Bucha- rest, Romania, July 29 – 31, 2006.
5 刘秋生,李红贵.基于事件驱动 SOA 架构的企业应用集成模式研究.中国管理信息化, 2009,4:67 – 69.
6 赵贝尔.综合数据业务平台(IDP)的设计及核心功能实现[硕士学位论文]. 北京:北京邮电大学, 2007.